An Automatic Method of Above-water Iceberg 3-D Profiling Based on Autonomous Surface Craft (ASC) Using Surface Images and LIDAR

by

© *Yahui Wang*

A thesis submitted to the

School of Graduate Studies

in partial fulfilment of the

requirements for the degree of

Master of Science

Department of *Physics and Physical Oceanography*

Memorial University of Newfoundland

November 2015

St. John's Newfoundland

Abstract

This thesis reports on a novel method to build a 3-D model of the above-water portion of icebergs using surface imaging. The goal is to work towards the automation of iceberg surveys, allowing an Autonomous Surface Craft (ASC) to acquire shape and size information. After collecting data and images, the core software algorithm is made up of three parts: occluding contour finding, volume intersection, and parameter estimation. A software module is designed that could be used on the ASC to perform automatic and fast processing of above-water surface image data to determine iceberg shape and size measurement and determination. The resolution of the method is calculated using data from the iceberg database of the Program of Energy Research and Development (PERD). The method was investigated using data from field trials conducted through the summer of 2014 by surveying 8 icebergs during 3 expeditions. The results were analyzed to determine iceberg characteristics. Limitations of this method are addressed including its accuracy. Surface imaging system and LIDAR system are developed to profile the above-water iceberg in 2015.

KEY WORDS: Iceberg; surface imaging; 3-D model; automatic; LIDAR

Acknowledgements

This thesis includes the research I have done in Memorial University from 2013 to 2015. First of all, I want to express my sincere gratitude to my supervisor Professor Brad deYoung and co-supervisor Ralf Bachmayer, who provided me the opportunity to do research in such an interesting discipline. Their insightful advice helped me a lot in my research. Also, I have been impressed with their dedication in doing research, which will influence me in my whole life.

Secondly, I would like to thank my colleagues in Autonomous Ocean System Laboratory: Neil Riggs who helped me to arrange field trips and order devices; Mingxi Zhou who provided me with lots of materials to assist me to start with my project, and gave me suggestions in data processing algorithm; Zhi Li who helped me with the field works regarding the ASC and the compatible software design; Haibing Wang who helped me with the mechanical design of my devices; Federico Luchino who helped me with some electrical design. Samantha Banton and Robin Matthews from Department of Physics and Physical Oceanography, along with Rong Sheng, a work term student in AOSl, helped me in the field works with the manual data collection. Further more, I want to give my extreme gratitude to Dr. James Munroe, who helped me with the volume intersection algorithm and OpenSCAD.

Finally, I want to thank my supportive family: my dear parents and my boyfriend Yang Wu. They gave me the power to overcome every obstacle I came across.

Contents

List of Figures

List of Abbreviations

ASC	Autonomous Surface Craft
AOSL	Autonomous Ocean System Laboratory
AHRS	Attitude Heading Reference System
BBB	Beagle Bone Black
CAN	Control Area Network
COP	Center of Projection
IOSS	Ice Ocean Sentinel System
LRF	Laser Range Finder
NRC	National Research Council
NSERC	Natural Sciences and Engineering Research Council
PERD	Program of Energy Research and Development
SFM	Structure from Motion
SFS	Shape from silhouette
SIFT	Scale-Invariant Feature Transform
UTC	Coordinated Universal Time

Chapter 1

Background

1.1 Motivation

Icebergs along the Newfoundland and Labrador coast threaten offshore facilities and activities, including oil and gas production platforms, collection and offloading systems, exploration schedules and marine transportation. The icebergs near Newfoundland and Labrador come from the glaciers of western Greenland and the glaciers on islands in Canada's Arctic area. The edges of glaciers break off and slip into the ocean to become icebergs. Every year around 40,000 medium- to large-sized (Table 1.1) icebergs break off from Greenland glaciers, and around 400-800 icebergs, the numbers varies from year to year, make it as far south as St. John's.[32] The icebergs near Newfoundland and Labrador normally don't have very large volume and most of them are medium- and small-sized (Table 1.1), the icebergs are always on the move at a significant drifting speed around 0.5 to 0.7 knots in the Grand Banks areas [32], for which reason the development of iceberg management needs to be paid attention by both the government and oil and gas companies.

Table 1.1: Iceberg Size Classification [31]

Iceberg Type	Code	Mass(T)	Height(m)	Length(m)
Growler	GG	500	$<$ 1 m	$<$5 m
Bergy Bit	BB	1,400	1 - 5 m	5 - 15 m
Small Berg	SB	100,000	5 - 15 m	15 - 50 m
Medium Berg	MB	750,000	15 - 50 m	50 - 100 m
Large Berg	LB	5,000,000	50 - 100 m	100 - 200 m
Very Large Berg	VB	$>$5,000,000	$>$100 m	$>$200 m

Iceberg management systems have been developed, implemented, and refined by research activities over the past 40 years to improve marine safety and protect offshore installations and marine transportation from iceberg threats. A survey of iceberg detection techniques and capabilities was conducted by the National Research Council (NRC) Canada to develop a strategy to perform iceberg management [12]. East Coast Ice Engineering Issues studies sponsored by the Program of Energy Research and Development (PERD) administered by NRC Canada give practical solutions to ice problems and excellent data resources [31]. The aims of iceberg management could be summarized as: (1) ensuring the safety of platform operation in the environment for which it was designed; (2) reducing risk to personnel and assets over and above design requirements; and (3) minimizing disruptions to drilling or production operations [31]. Effective ice management should fulfill the tasks of iceberg detection, decision making, and altering the icebergs path by towing or other means if necessary. Iceberg detection is generally undertaken by satellite imagery or marine radar, and the measurements of the size of the above-water icebergs mostly rely on these two methods, even though the resolution is not sufficient for further operations. However, high resolution models of icebergs are required since they include significant

information required for decision making. This can be seen from the C-CORE Iceberg Decision Making Toolbox [9], in which the independent variables, iceberg location and features of the iceberg influence the decisions of when, where and which icebergs to tow. These variables are size, mass and shape, and the structure operation and corresponding T- time (total time required to complete operation), ice load capacity of the structure, tow resource locations and environmental conditions [9]. Dunderdale's Iceberg Management Planning Aid (IMPA) is designed to improve iceberg towing effectiveness through prediction of expected drift outcomes for a given input of iceberg size, shape, drift velocity and direction [8].

For above-water iceberg observation, traditional techniques include photography, radar or other surveying techniques, and ground penetrating radar. Although they can give some information about draft and mass, they are not feasible approaches to get the shape of the iceberg [11]. Canatec produced a comprehensive database that integrated the three-dimensional shape and geometry of icebergs with records that contain 3-D data observed on the Grand Banks of Canada [6]. Fugro [14] announced at The 2014 Arctic Technology Conference an above-water imaging techniques (3-D photogrammetry) to generate 3-D models of icebergs, but no further report could be found.

1.2 Problem Statement

Each year a large number of icebergs in the North-West Atlantic, making it the place with the most ship-iceberg collisions in the world. As more and more new reservoirs being discovered, there will be more offshore operations involved in this iceberg threatened area. The icebergs are one of the most important concerns for oil and gas industry in this region.

However, surveying icebergs can be extremely dangerous using traditional techniques, because it is possible that the icebergs could break apart suddenly. It is not safe for the personnel or the vessels to survey the icebergs from a short distance. Also, new autonomous technique of getting the accurate parameters of the interested iceberg in a short time is required to make the iceberg under control and not damage the offshore architectures.

To overcome these problems, Autonomous Ocean System Laboratory (AOSL) has developed an autonomous surface craft (ASC) to conduct iceberg surveys in order to guide the decision-making task. Fig 1.1 shows the ASC used in this thesis to do the experiments, as designed and built by Li and Bachmayer. [18]

Figure 1.1: A catamaran-type ASC developed by Memorial University, with 1.5 m in length, 1 m in width, 1.5 m in height[18]

Understanding the 3-D content of an object based on 2-D images is a central problem in computer vision, and many approaches have been proposed for reconstructing 3-D models when 2-D images are available. There are some open source 3-D modelers using Structure from Motion (SFM) that have been developed in recent years, such as AutoDesk 123DCatch, VisualSFM, Agisoft, and Bundler. These software packages cannot be used on the ASC because they cannot be run under completely autonomous conditions. Furthermore, the software uses SFM techniques, which have some limitations in the task of 3-D iceberg model reconstruction. Simply speaking, SFM techniques begin by assuming a perspective projection model (Fig 1.2). Here, for example, three 3D feature points are projected onto an image plane with perspective rays originating at the center of projection (COP), which would lie within the physical camera. The focal length (f) is the distance from the COP to the image plane along the optical axis. Therefore, these three points' location can be estimated in real world and produce a point cloud. [15]

One of SFM's limitation is that its key-point-based reconstruction only works well with textured surfaces, such as a white wall and uniform colored objects. That is because the Scale-invariant feature transform (SIFT) which is widely used in SFM techniques is not adequate to extract enough information from images with irregular shapes and shadings, for example, iceberg images. In addition, the cloud points produced from SFM is not closed, and these points are far from enough to build an accurate 3-D model to estimate its volume if too few features are detected. Therefore, it is hard to realize the function of estimating the volume of the iceberg if using SFM. What's more, the principal disadvantage of the SFM technique is that the mathematical process is not evident, and not enough research has been done or published to convince researchers of its accuracy and reliability [28].

To quantitatively measure the icebergs shape and size using the ASC as the platform,

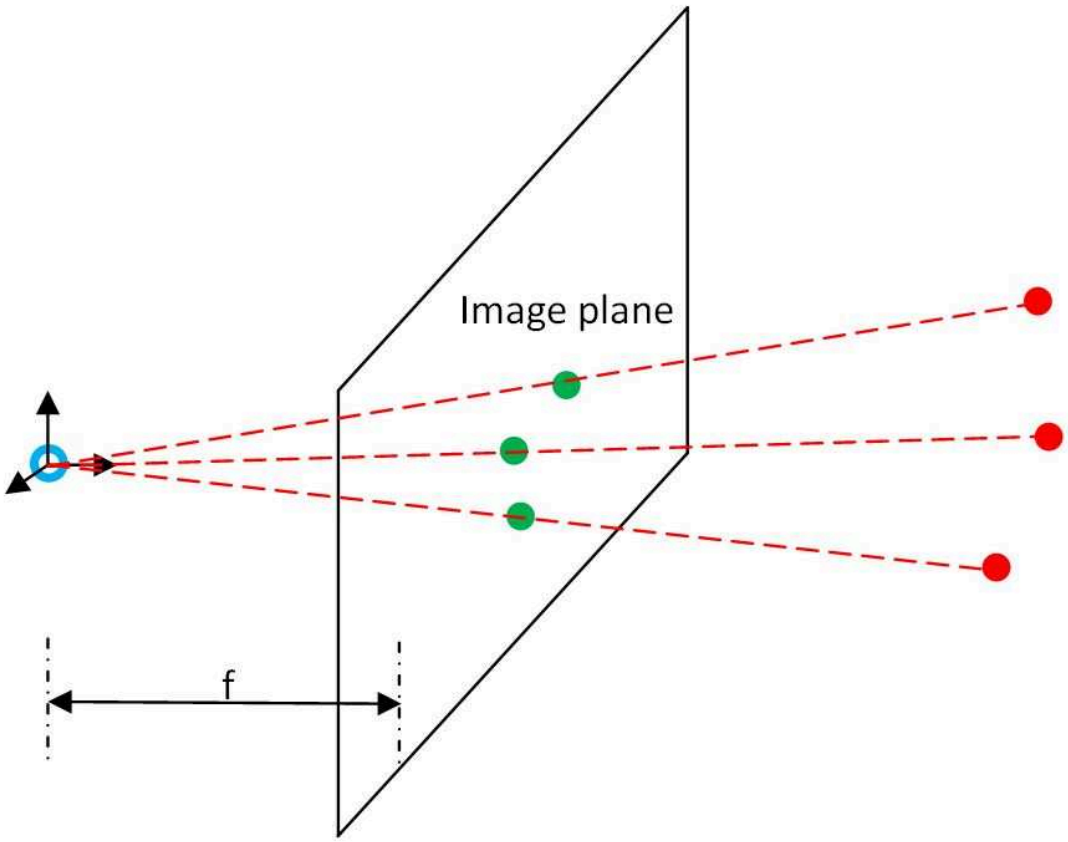

Figure 1.2: Perspective projection model in Structure from Motion (SFM)[15]

an effective method, including hardware system and software algorithm, need to be developed. Specifically, these requirements must be met: (1) producing 3-D models fast; (2) building 3-D models and estimating parameters accurately; and (3) working on the ASC autonomously.

1.3 Structure of this Thesis

In this thesis, the author reports on a technique for constructing 3-D iceberg models and estimating their volumes by using 2-D photographs and other collected data (such as GPS, distance, angle) on the basis of using the ASC. This work is one part of the Ice Ocean Sentinel System (IOSS), which is endeavoring to provide critical ocean information and data products for decision-making needs in frontier, harsh and ice-prone environments.

In Chapter One, the motivation and background of this study are introduced. In Chapter Two, the main methodologies, including volume intersection, occluding contour finding, feature matching, and parameter estimation, used in this algorithm are explained. This chapter includes a literature review of previous work. For methodologies developed in this thesis, such as iceberg parameter estimation, some supporting equations and figures are also included.In Chapter Three, the development of the algorithm is presented. The first step is the manual method, then becomes a partial automatic method, and currently it can theoretically work automatically on the ASC (this algorithm can work independently on computer, but due to time and hardware limitations, it hasn't been transferred to the ASC yet). At each stage, different experiments are designed to improve the algorithm.

In Chapter Four, the entire method is explained in step by step detail. In the end, a flow chart are provided for easier understanding. In Chapter Five, data from the PERD Iceberg

Shape Database are used to estimate the resolution of this algorithm. Chapter Six shows some results from iceberg surveys during the iceberg season of 2014 and 2015, which verifies the feasibility of the algorithm. In Chapter Seven, LIDAR is used in the profilling of above-water icebergs. The LIDAR system design is presented and the results from field surveys. In Chapter Eight, conclusions are provided as well as suggestions for future work.

Chapter 2

Methodology

2.1 Volume Intersection Algorithm

As stated in the previous chapter, the SFM method is not appropriate for the task of building a 3-D model of the above-water icebergs because of the limitations. In this thesis, the Shape from Silhouette (SFS) , which is also called volume intersection, will be introduced to perform the function to build 3-D models from 2-D images, since the method is suitable to reconstruct 3-D shapes for irregular, solid objects easily and rapidly.

Some approaches can be used to reconstruct the 3-D shape of an iceberg from its 2-D images using SFS. Martin [22] first presented a method to get volumetric description from multiple views. Many other researchers reconstructed surface volume by performing planar motion under orthographic projection on solid objects [17] [10] [25], and the method is referred to as volume intersection algorithm, which could be described as follows [3]:

> *"The word silhouette indicates the region of a 2-D image of an object O which contains the projections of the visible points of O. If no a priori knowledge about O is available, all the information provided by a silhouette Si is that O must lie in the solid region of space Ci obtained by back-projecting Si from*

the corresponding viewpoints Vi. If n silhouettes are available, they constrain O within the volume Rn:

$$n = \bigcap_{i=1}^{n} Ci \tag{2.1}$$

”

The most critical information needed here is the silhouette extracted from the 2-D images since it has been recognized as the most effective clue to understand shape. These silhouettes are extruded by sweeping the silhouette along either a line parallel to the viewing direction (Fig. 2.1) or a cone obtained by back projection from a viewpoint. The volume intersection can be realized from all different directions for solid objects. However, in this project I initially constrain the viewpoint as parallel to the sea- surface since I am using a vehicle on the sea-surface to collect images of icebergs.

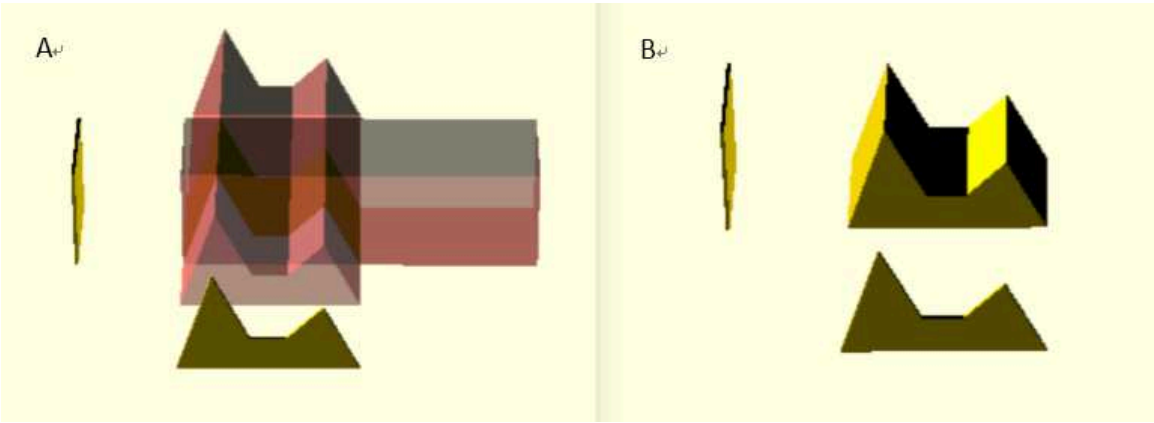

Figure 2.1: Simplified volume intersection algorithm: A is is the extrusion of two different silhouettes (the front view is an irregular shape, and the left view is a square); B shows the isometric view of the object after intersection.

The occluding contours can be extruded along the parallel projection lines and the re-

sulting volumes can be intersected and extracted. Therefore, a 3-D object that gives the same silhouette of the captured iceberg from the specific viewpoint can be generated. With more intersections, the precision can be improved. The resulting object is called a visual hull. This is the idea behind constructing a 3-D model of an iceberg from its 2-D images. When an object is convex, a good approximation of the original object can be obtained (Fig 2.2). However, the visual hull is not an exact reconstruction of the original iceberg, even when performing an infinite number of intersections. This is because silhouettes are not sufficient to determine the 3-D shape when the object has a concave form [3], as can be seen from Fig 2.3. The 3-D model of the iceberg mentioned in this thesis is the visual hull, if not specified. For the specific characteristics of icebergs, the non-convex polygon can only be seen in the top view, as other views could be shown in the occluding contours.

Although the SFS has the significant disadvantage that the resulting shape size will be significantly larger than its real size when the shape is a non-convex polygon, this problem can be minimized by adding extra measurements to resolve the concave form of the object. In this case, an additional sensor, a laser range finder, is deployed in order to improve the representation of the object. The laser measurements serve to fill in the hollow shapes of the iceberg.

After the introduction of the choice of the core methodology for iceberg shape reconstruction, there remain several issues to address including: extraction of the silhouette, data collection, and data processing.

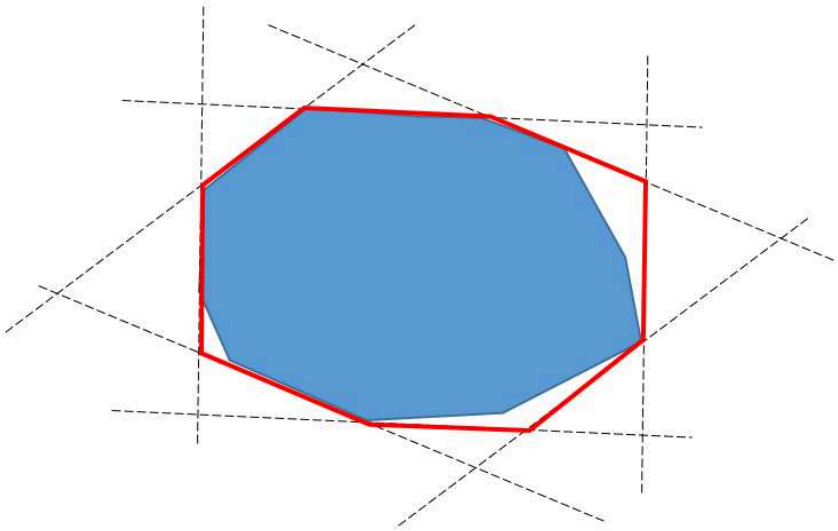

Figure 2.2: Visual hull for a convex polygon

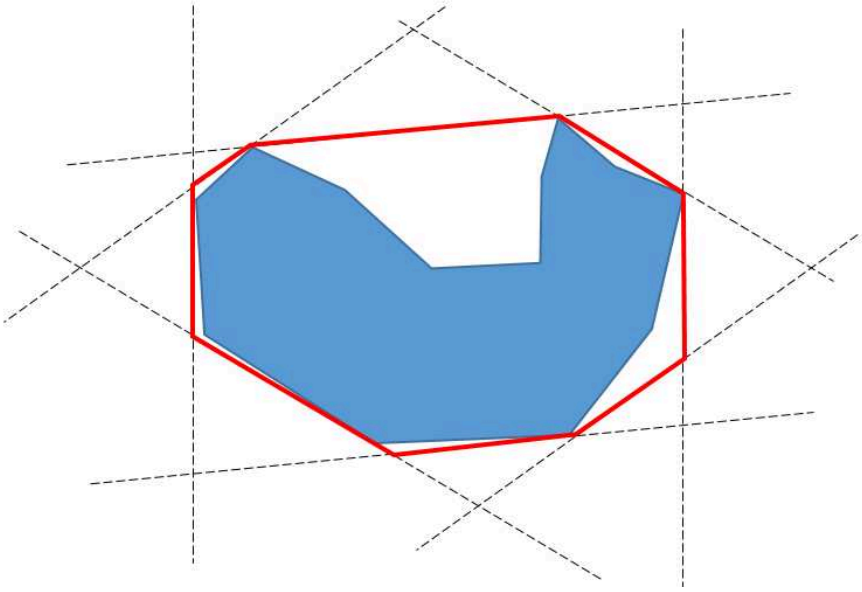

Figure 2.3: Visual hull for a non-convex polygon

2.2 Occluding Contour Finding

As the first step, I extract occluding contours, which provides a partial representation of the true iceberg shape, from the images of the icebergs. However, contour extraction is significantly different from other standard computer vision algorithms since the icebergs shapes and shadings are irregular. In addition, the iceberg color is usually a gradient that might be very similar to the color of the clouds or the sea itself.

Roberts [27], Sobel [13] and Prewitt [24] proposed edge detection by convoluting differential operators with gray scale images. In order to improve the performance, more recent approaches take color and texture information into account, and learning techniques are used for cue combination [21] [19]. However, these methods are not suitable for natural images with large gradients, such as iceberg images. Ren [26] found that combining global cues can improve the operation by reducing clutter and completing contours, and he proved that this method can perform better than SIFT, which is the widely-used algorithm used for extracting information from 2-D images in SFM [7] [20]. Based on this previous work, a state-of-the-art algorithm used for contour finding was developed by the UC Berkeley Computer Vision Group [4]. The algorithm couples multi-scale local brightness, color and texture cues to build a powerful globalization framework using spectral clustering. To reduce the computational complexity of later stages and to clearly capture object-part relationships (for example, an image of a person contains a face region nested within a region corresponding to the entire body), the result can be shown in a hierarchical segmentation tree in different intensities of boundary contours [20]. Hierarchical segmentation is the most ideal method for iceberg contour finding because the method shows the difference between the whole iceberg object and iceberg details, which can help to find the whole

iceberg object easily. In the end, the output yields a group of closed contours, which can be seen as either segmentation or closed boundaries.

For a given threshold, the output of hierarchical segmentation can be binarized to delete the unnecessary contours. The researcher assumes that the whole iceberg is always inside an image otherwise the image was not be considered, and then generated the single closed iceberg contour by extracting all the regions that cannot be reached by filling in the background from the edge of the image.

2.3 Parameter Estimation

2.3.1 GPS Location

The ASC can record its real time GPS position and orientation using its GPS and attitude heading reference system (AHRS). I also used a laser range finder to measure the real time distance away from the iceberg. This is done with the assumption that the iceberg does not shift much during the time of data collection, as the survey is done within ten minutes. To combine all this information, I use equations 2.2 and 2.3 to calculate the GPS location of the iceberg.

$$I\_lat = V\_lat + Dx * 360/(2 * \pi * R) \tag{2.2}$$

$$I\_lon = V\_lon + Dy * 360/(2 * \pi * R * cos(I\_lat)) \tag{2.3}$$

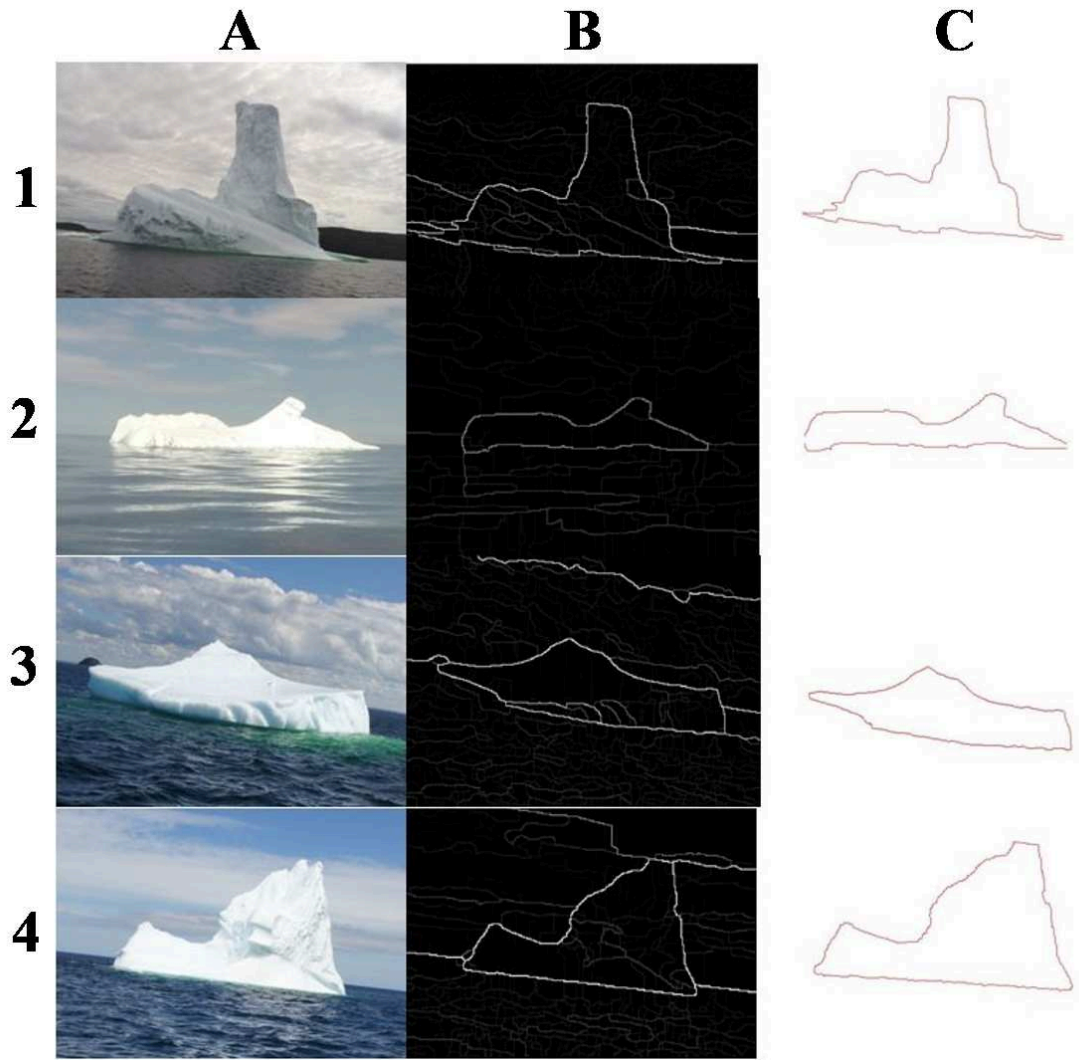

Figure 2.4: Occluding contour finding. A column are original images, B column are the result produced by hierarchical segmentation[4], and C column are the final contours of the surface images. These four icebergs in different rows were observed on Jul 30th, 2014 near Twillingate, Newfoundland. The waterlines in these figures are not always horizontal because of the movement of the vehicle.

In these equations, R denotes the radius of the earth assuming that the earth is spherical. V_lat, V_lon denote the real-time latitude and longitude of the vehicle. Dx and Dy are the east and north components of the distance measured by the laser range finder. I_lat and I_lon denote the latitude and longitude of points on the surface of that iceberg.

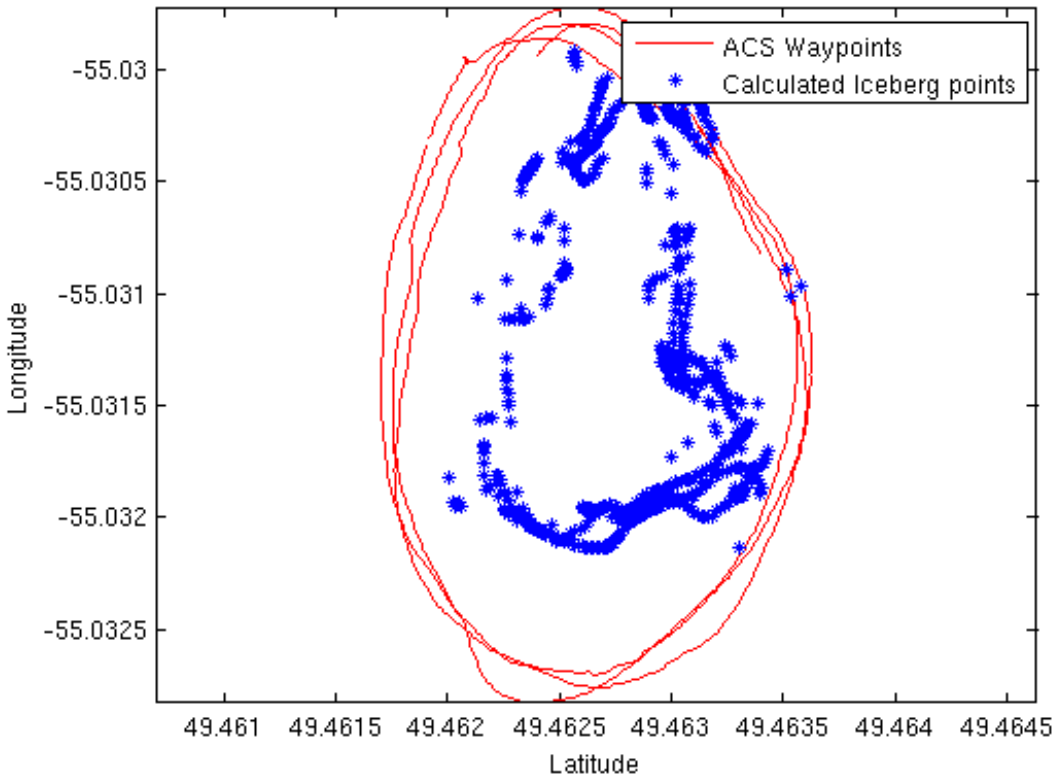

Figure 2.5: GPS location of the waypoint and the iceberg on July, 2014

Fig 2.5 shows the results of one iceberg observed on Jul 30th, 2014. The red line is the route of ASC, while the blue stars are the points on the surface of iceberg. Its shape is calculated from the GPS and heading of the vehicle, and the distance between the iceberg and the vehicle. The laser range finder is only one dimensional, just giving distance to

the iceberg, but with comparable high sampling frequency (1Hz), so I can get the two dimensional top view shape of the iceberg using the distance between the laser range finder and the bottom of the iceberg, when the device point to the bottom of the iceberg. This shape can later be used to partially resolve the visual hull shortcoming of the 3-D shape estimation process as discussed earlier.

The iceberg size derived from GPS laser range data can be used to provide a first size estimation of an iceberg. By overlaying the computed iceberg outline and the ship position information, a first reality check are provided on the raw data. The GPS information can be converted into earth coordinates (Fig 2.5), making the combination of the result of above water and underwater portion [33] of the iceberg theoretically available, and thereby enabling a determination of the complete shape of the iceberg.

2.3.2 Drifting Parameter Estimation

Most of the icebergs observed near the coast were not grounded. They drifted as they were observed with the ocean currents. If the speed of the iceberg's movement is not negligible, the drifting of the icebergs could make a significant influence on the final result. Therefore, determining the drifting parameter, for example, the velocity (including the drifting speed and the rotation angle), is an essential part in the above-water iceberg 3-D modeling.

Since the images and data collection could be controlled in a short period of time, I made a simplified assumption that the iceberg moves with a constant velocity. By plotting the route of the ASC using GPS locations, it is clear that the iceberg was drifting (Fig 2.6). As seen from this figure, it is hard to recognize anything without some consideration of the iceberg drift.

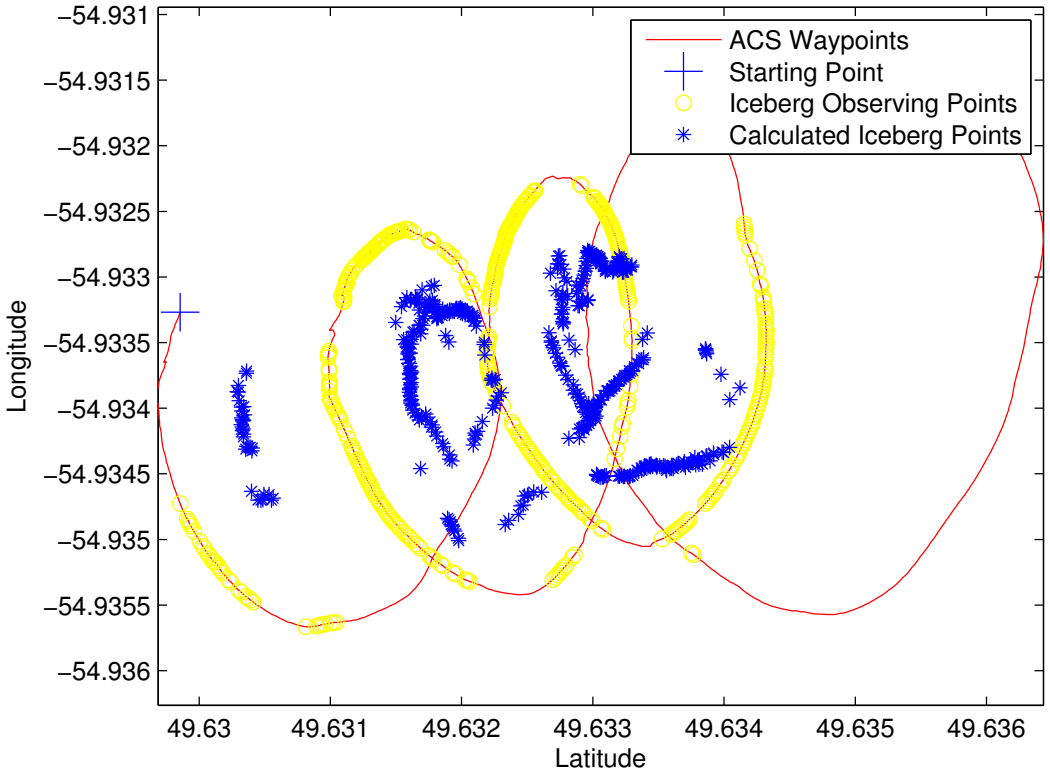

Figure 2.6: GPS locations of a drifting iceberg. This iceberg was observed on Jul 30th, 2014. The real-time iceberg points (blue *) could be calculated from the observing points (yellow o).

The analysis of the drift can be made from two different perspectives:

1) Horizontal velocity. The horizontal velocity could be estimated through the route of the ASC because the distance between the iceberg and the ASC tends to be stable, as our experiment disign is to go around the iceberg several time at a certain distance. As the iceberg drifts with the ocean currents, the ACS will also go in roughly the same direction. The laser range finder is sampled at 1Hz. Based on the number of points, how long the iceberg takes to drift such a distance can be then estimated. Therefore, the horizontal velocity could be calculated using the collected data. By a rough calculation of the drift velocity, an iceberg coordinate can be set up instead of the earth coordinate (in the earth coordinate the earth is stationary, while in the iceberg coordinate the iceberg can be seen as stationary). As we know the distance between the ASC and the iceberg, which is measured by the laser range finder, the top-view shape of the drifting iceberg can be plotted in iceberg coordinates.

2) Angular velocity. The images collected at the same heading with different GPS locations will be compared and a correlation coefficient will be calculated. If the correlation coefficient is not high enough, the closest image will be compared with the original image, until the highest coefficient is found. Those two images producing the highest coefficient will be considered to be captured from the same angle. The angular velocity can accordingly be estimated.

Fig 2.7 is the top-view shape of this drifting iceberg after the drifting parameters correction. Now it is easy to recognize the top-view shape of this iceberg.

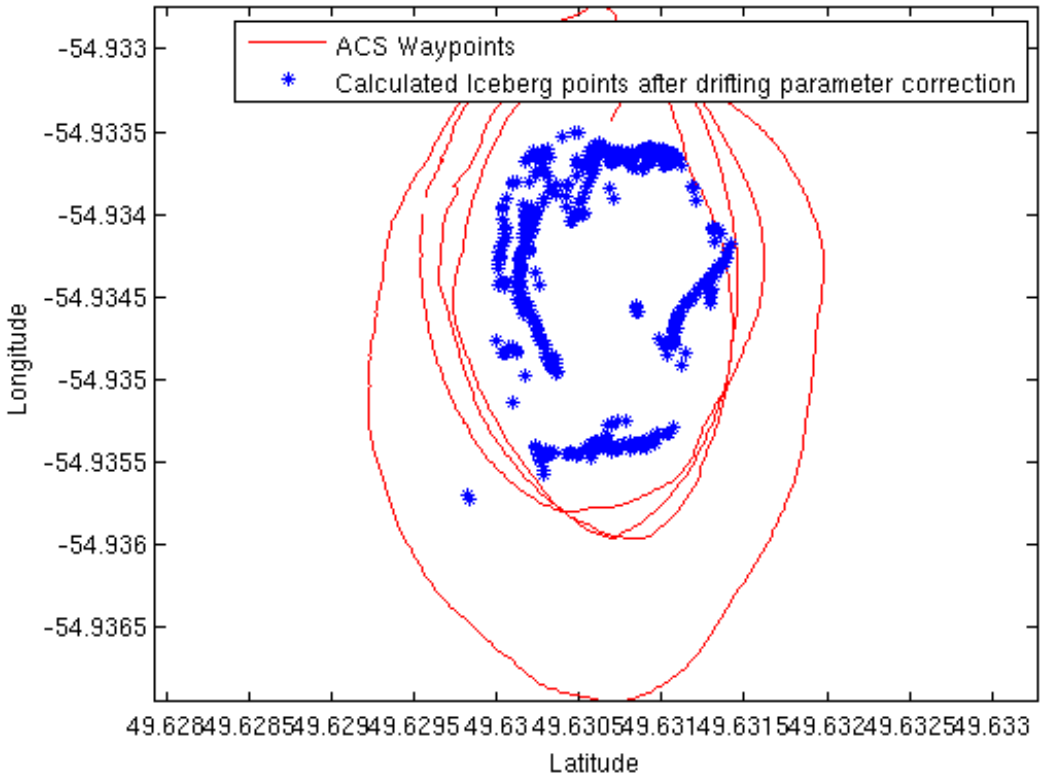

Figure 2.7: GPS location of drifting iceberg after correction (observed on July 2014)

2.3.3 Size Calibration

Using the infinite focus camera mode and the known distance between the lens and object, the pixel count can be used in the image frame to determine the size of the iceberg. However, since icebergs are not flat, the lens-iceberg distance varies. Therefore, a new way has to be used to calibrate the size of icebergs.

The general shape classification of icebergs is shown in Table 2.1.

Table 2.1: Iceberg Shape Classification [31]

Iceberg Shape	Description	Illustration)
Tabular	Horizontal or flat-topped with length to height ratio of 5:1 or more	Fig 2.8.A
Blocky	Steep precipitous sides with near horizontal top and length to height ratio of less than 5:1	Fig 2.8.B
Domed	Smooth round top	Fig 2.8.C
Dry Dock	Eroded such that a large U-shaped slot is formed with twin columns or pinnacle slot extends into the water-line or close to it	Fig 2.8.D
Pinnacle	One or more large spires or pyramids dominating overall shape	Fig 2.8.E
Wedge	A tabular iceberg which has altered its position of stability so that is now appears tilted, resembling a wedge.	Fig 2.8.F

Assuming that the peak can be seen from all angles, and that it does not change significantly with the shifting motion of the iceberg, the height remains almost unchanged. Compared with the base of the above water iceberg, the peak has almost no thickness so

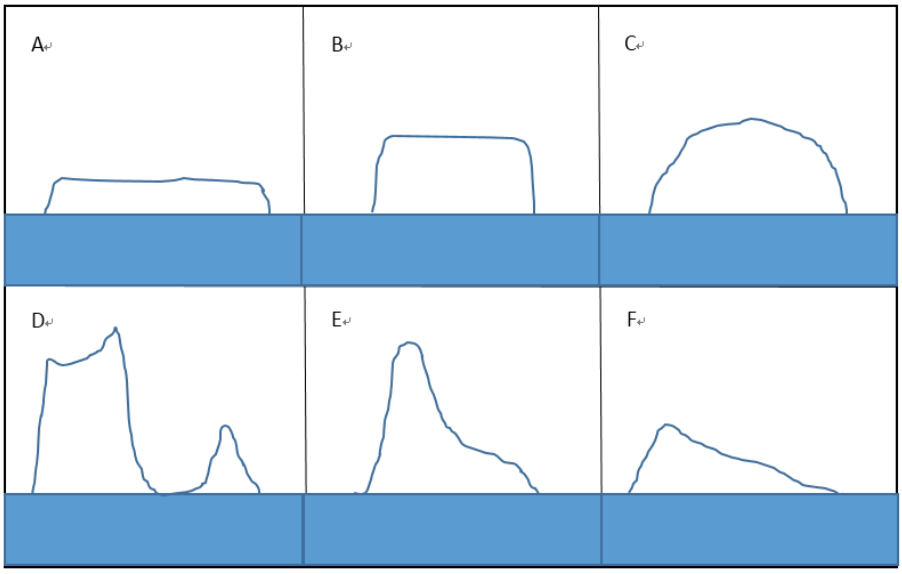

Figure 2.8: Iceberg Shape Classification

the distance is more uniquely defined.

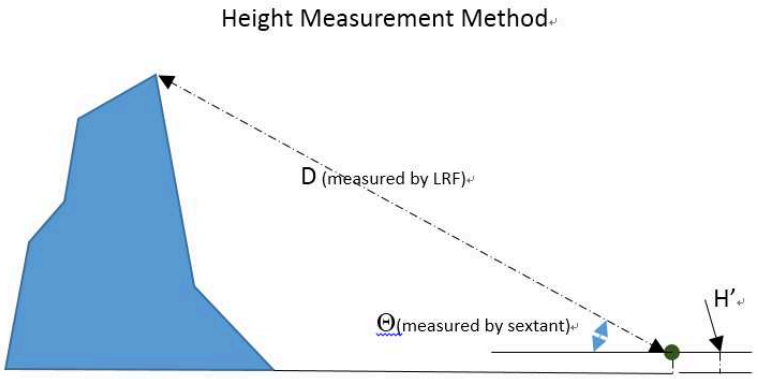

Figure 2.9: Height measurement method. D denotes the distance between the vehicle and the iceberg, which is measured by the LRF; θ denotes the angle between the sea surface and the peak of the iceberg. a) Tabular. b)Blocky. c)Domed. d)Dry Dock. e)Pinnacle. f)Wedge.

Fig 2.9 shows the method of height measurement. θ' denotes the angle between the sea surface and the view angle of the iceberg's peak. Assuming the height of the sensor (H') is negligible, θ equals to θ' approximately. The height of the iceberg (H) can be calculated easily based on the same assumption. The measurements taken from different viewpoints can be denoted Di and θi. The height accuracy ($\bar{H}$) is enhanced in equation 2.4 and 2.5 by making measurements from different view-angles and using the average height.

$$Hi = Di * sin\theta \tag{2.4}$$

$$\bar{H} = \sum_{i=0}^{n} Hi \tag{2.5}$$

To this extent, the pixel of original image and the focal length were not critical, so a fixed height pixel for all silhouettes was used for convenience to perform the volume intersection algorithm, while changing the whole silhouette proportionally. The resulting 3-D shape was calibrated with the real world size (or height). This method can only apply for the manual collection experiments since sextant is used.

For the automatic collection, one possible way is to determine the size of the iceberg using the LRF. The iceberg's shape and GPS location could be measured and analyzed using equation 2.6 and 2.7, which can then be transformed to meters from earth coordinate. L denotes the length of the iceberg along the latitude, and W denotes the width of the iceberg along the longitude. Lon denotes the measured longitude, and lat denotes the measured latitude. As the iceberg is small compared with the earth, I set LON (the average of lon) to

be the base longitude. R is the radius of the earth assuming the earth is a sphere.

$$L = (max(lat) - min(lat))/360 * 2 * \pi * R * cos(LON) \tag{2.6}$$

$$W = (max(lon) - min(lon))/360 * 2 * \pi * R \tag{2.7}$$

Fig 2.10 shows the top view of the iceberg model, in which the blue solid is the 3-D model of iceberg (units in meters) calibrated by manual height measurement, while the red stars denote the GPS dots transferred to meters. Those two results are to the same length scale, which confirms that it is reasonable to use the LRF to estimate the length and width of the icebergs, and then calibrate them without the manual height measurement.

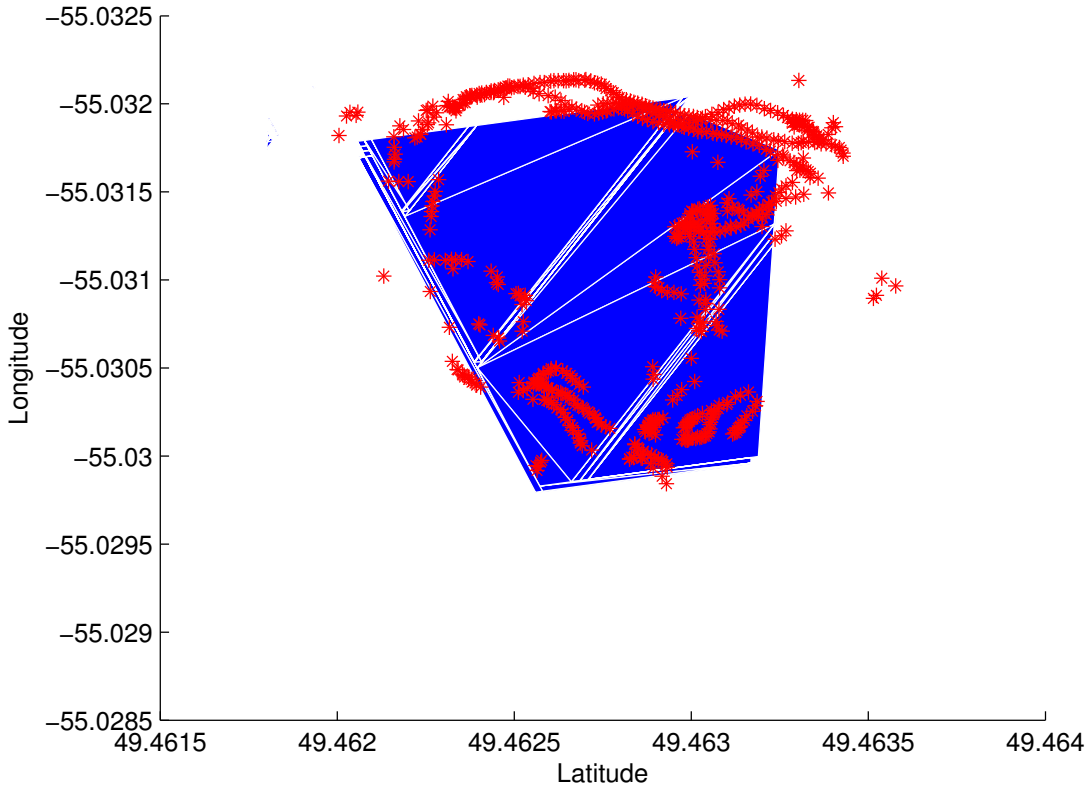

Figure 2.10: Length (units in meters) transferred from GPS location. The red stars are calculated from the LRF measurements, and the blue shape is the model that I built.

Chapter 3

Experiment Design

3.1 Manual Data Collection

This manual data collection method was the first step to design and test the algorithm, which is the basis of the automatic method. Fig 3.1 shows the basic idea of the manual collection method. Typically personnel go around the iceberg in a small inflatable craft - called Narwhale (the name of a fast rescue craft). The red markers indicate the points where data was collected, including images, GPS locations, heading of vehicle, environmental parameters, etc. I started with this approach to control the sampling platform because the integration of surface craft is not fully operational by then.

The manual method also serves as a backup for the automated data collection method which at the first stage might present discrepancies during expeditions. Data collection around the iceberg is required for analysis. The choice of collection locations on this circle should be distributed around the iceberg, but those location points can also be arbitrary as the GPS location of the points have been recorded. The number of locations depends on

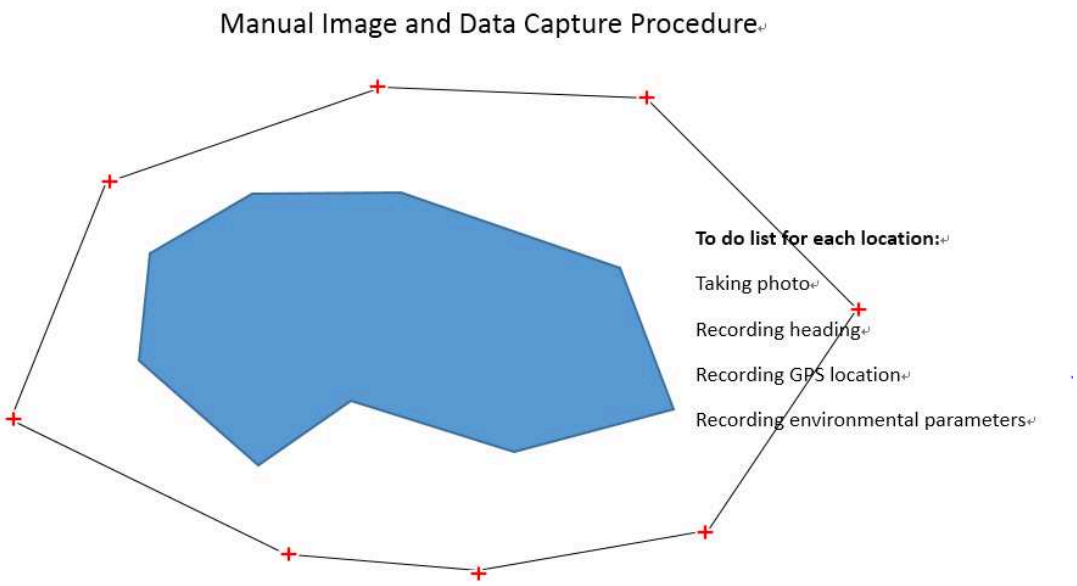

Figure 3.1: Manual Data Collection Method

the size and shape of the iceberg. In general, accuracy should increase with more measurements. If only a few locations are taken, the resolution could be low. However, having too many locations is also not desirable because the measurement errors from all locations could add up, and makes later processing more complicated. For each location, a complete data point will consist of an image, its GPS position, and camera orientation for redundancy purposes manually measured. Different data points are taken for different sized icebergs, such as 5-6 points for a small-sized iceberg, and 9-10 for a medium-sized iceberg. But the accuracy depends on how many points are used no matter the size of the iceberg. Typically 9-10 points are a good selection, which will be explained in detail in the following chapters.

Note that the height measurement mentioned in 2.3.3 via hand-held laser range finder and sextant is only used in the manual method.

3.2 Partial Hardware Integrated Method

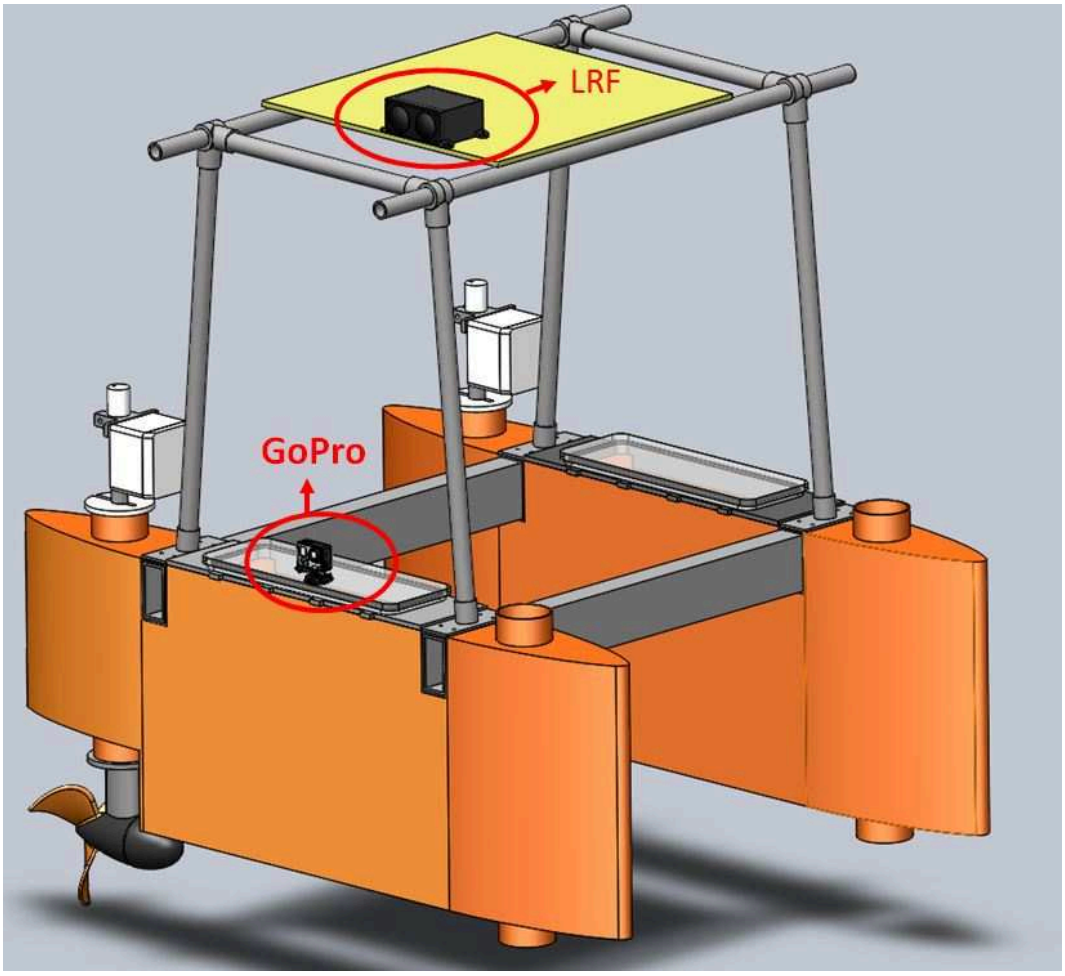

Figure 3.2: The placement of GoPro and LRF on the ASC in 2014

As stated in Chapter One, our goal is to use the ASC as a platform for doing an automated iceberg survey. Therefore, in the second stage of the experiments, the ASC is used as the primary data collection platform. The ASC is suitable to perform scientific tasks in harsh environments, as it can be used in conditions where are unsafe for the traditional vessels. The ASC designed at Memorial University is integrated with multiple sensors to

collect data, such as GPS location, orientation, heading, and environmental parameters, including temperature, wind speed, and humidity. In the summer of 2014, I attached a GoPro HERO3+ Black edition camera to capture the 2-D images of the icebergs, and a 1Hz sampling laser range finder (Lightware SF03/XLR) was also placed on the vehicle to measure the real time distance away from the iceberg (Fig 3.1). Since not all devices were integrated during the iceberg season of 2014, UTC time (coordinated universal time) was used to synchronize all data collected from different devices. A time-synchronized field computer was used for data collection.

In this trial, the ASC went around each iceberg three times. Manual height measurements were also taken in this trial.

3.3 Hardware Integrated Method

As mentioned above, the camera and laser range finder were not physically connected on the ASC using hardware in the summer of 2014. However, integration is needed in order to reduce the error caused by the discrepancy from synchronization. In addition, fully integrating and then building a full data frame can be useful for future data processing, as the controller can get the data without manual transferring. This is the basis for the automation of the ASC.

3.3.1 Subsystem Controller

BeagleBone Black (BBB) is a low-cost, community-supported development platform. Its Linux operating system is used for three reasons: 1) it can control both the sensors and camera directly to collect data; 2) the software used in this thesis can be easily installed on

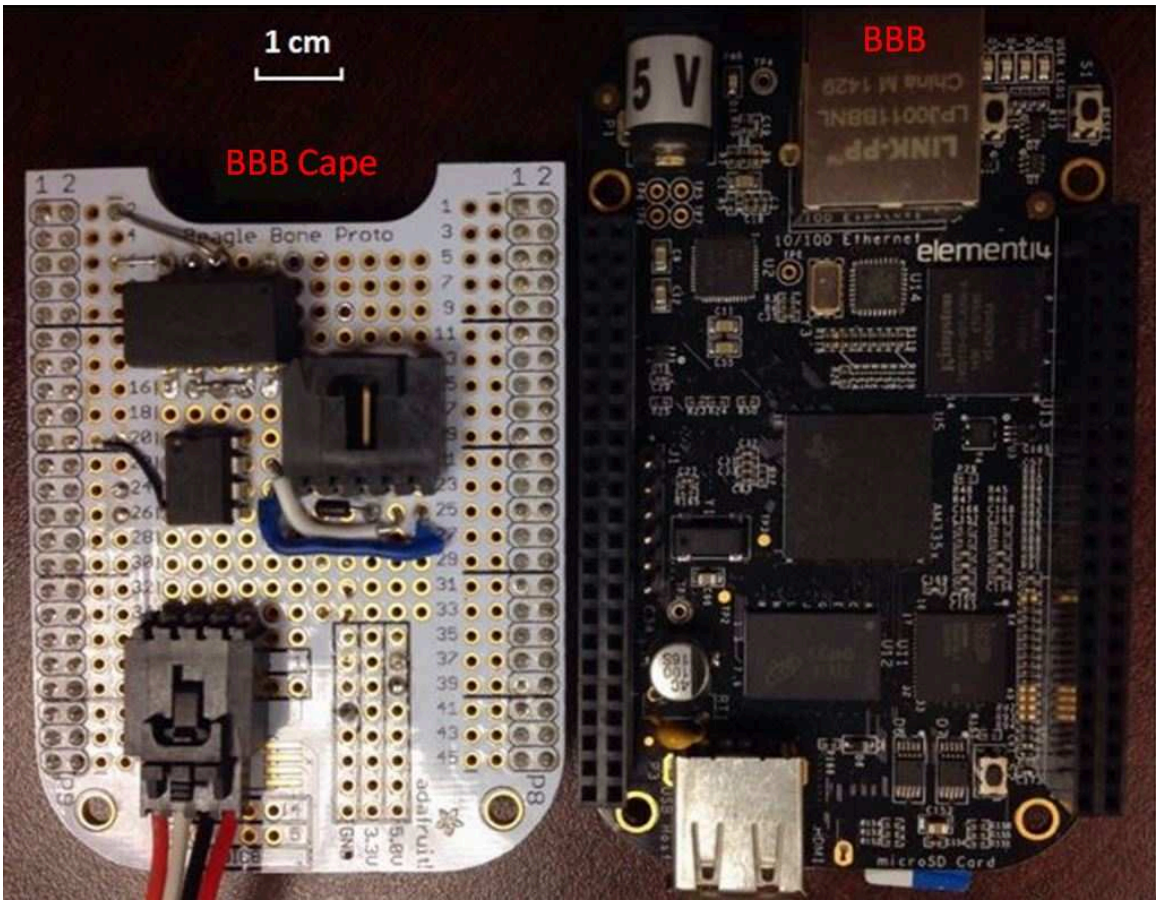

Figure 3.3: Micro-controller - Beaglebone Black: the right one is the BBB, and the left one is BBB cape, which are used to add external CANbus for the BBB

this operating system; and 3) shell scripts in the Unix environment can make the algorithm in this thesis work automatically on the ASC. Furthermore, its on-board processing unit features the AM335x 1GHz ARM Cortex-A8 processor, and also has the 512MB DDR3 RAM, 4GB 8-bit eMMC on-board flash storage, 3D graphics accelerator, NEON floating-point accelerator, as well as 2x PRU 32-bit microcontrollers [1]. As shown in Figure 3.3, the BBB takes the processor resources to the specific pins for convenient connections. The available resources for the BBB include the Ethernet, USB client for power and communications, USB host, HDMI and the serial port.

3.3.2 Camera

Figure 3.4: IP Camera - ATCi E37

ATCi E37 Network camera(Fig 3.4), is an outdoor camera with Weather-proof IP66-rated housing (Temperature: -20-50, Humidity: 90RH), was used to take iceberg images on the ASC. It has a waterproof connector, which can be used to integrate the IP camera with the BBB. Both network camera and the BBB have Ethernet. The network camera is deployed on a local area network (LAN) using a router and intended to be accessed by BBB.

Through HTTP protocol, the camera can simultaneously transfer streams to the BBB. The video streaming can be viewed through an Internet browser after IP address configuration.

The images can be taken using the ”wget” command.

3.3.3 Laser Range Finder

Lightware SF03/XLR laser rangefinder(LRF) (Fig 3.5) module is an OEM laser rangefinder with configurable update rates to provide fast distance measuring capabilities up to a range of 250 meters. The sampling rate that I use is 1Hz, which is compatible with the sampling rate of the Airmar PB200 Weather Station.

Figure 3.5: Laser range finder - Lightware SF03/XLR

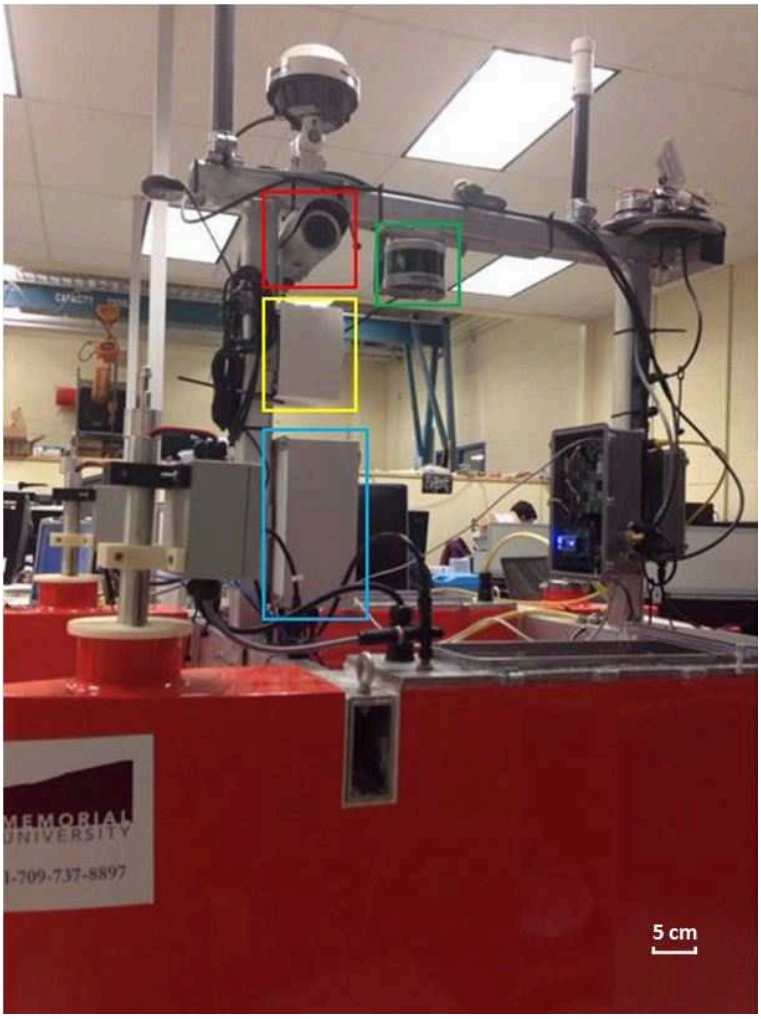

Figure 3.6: The placement of IP camera and LRF on the ASC in 2015: the red square highlights the IP camera, the yellow one the laser range finder, the green one the LIDAR, and the blue one the electrical control box.

3.3.4 Control Area Network (CAN)

The Controller Area Network (CAN) was originally used in the construction of the communication and control system for automobiles. It was developed in 1986. The ASC deploys CAN because of its specifications of broadcast communication, priority, and error capability. These specifications can make it easy to add more sensors to the control system.

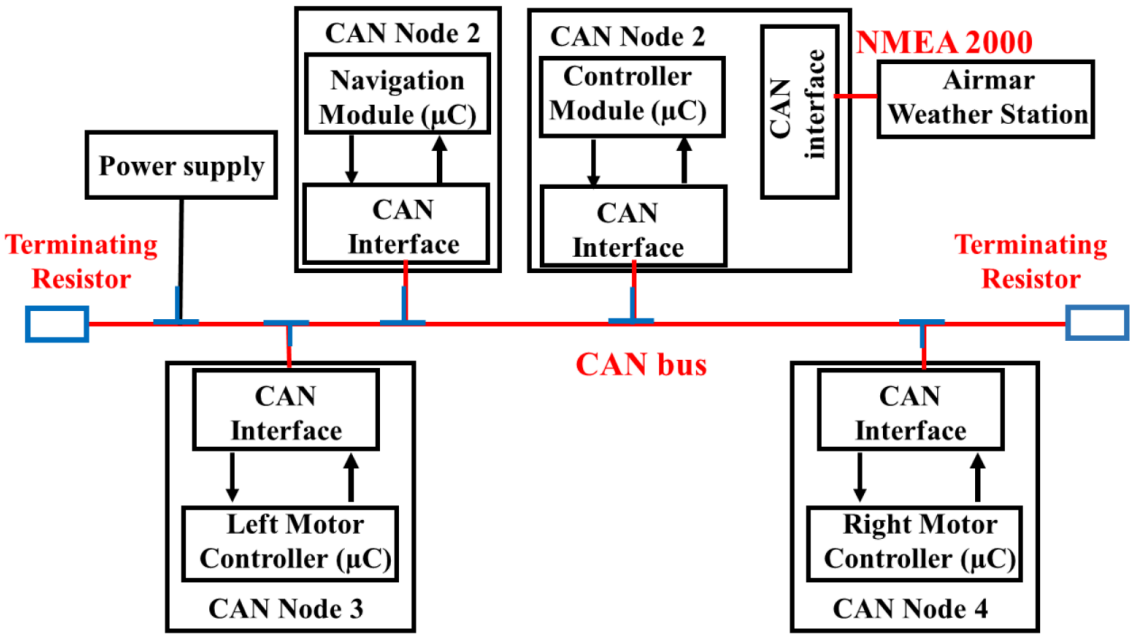

Figure 3.7: The ASC communication and power system based on the CANbus[18]

Fig 3.7 shows the ASC communication and control system structure using the CAN protocol. A number of nodes, including the two motors, navigation module, control module for the ASC, and the sensors (weather station) are connected into this system. As a multi-master communication protocol, all the CAN nodes were able to freely access the bus to send and receive messages. Our subsystem can also be connected and communicate with the ASC through the CAN bus.

For the subsystem of above-water iceberg 3-D modeling, the subsystem micro-controller can acquire CAN messages and save the sensors' data (i.e. latitude, longitude, heading) in the local SD card when receiving a message that the mission starts. At specific time points when the heading changes to a specific heading, the camera takes one image and saves it.

Chapter 4

Automatic Data Processing and Analysis

4.1 Image Pre-Processing

The first step in image processing is to do pre-processing on the images based on the ASC orientation collected from the AHRS on the ASC. Most of the time the water surface in the images are not horizontal because of the motion of the vehicle and the camera. Therefore we need to correct the images or contours on the basis of orientation (roll, pitch, and yaw) of the vehicle to make sure the contours could be used in the volume intersection.

One of the most widely used parameterisations for image rotation is the 3-2-1 Euler angles [5]. The 3-2-1 Euler angles provide an orthogonal matrix Q which connects a basis for fixed spatial frame {E1,E2,E3} and a basis for a body frame{e1,e2,e3}. The equation could be shown as (the whole equation is in the next page):

$$\{E1, E2, E3\} \xrightarrow{Q} \{e1, e2, e3\} \tag{4.1}$$

The orthogonal matrix Q could be demonstrated with the Euler angles (yaw-ψ, pitch-θ, roll-ϕ).

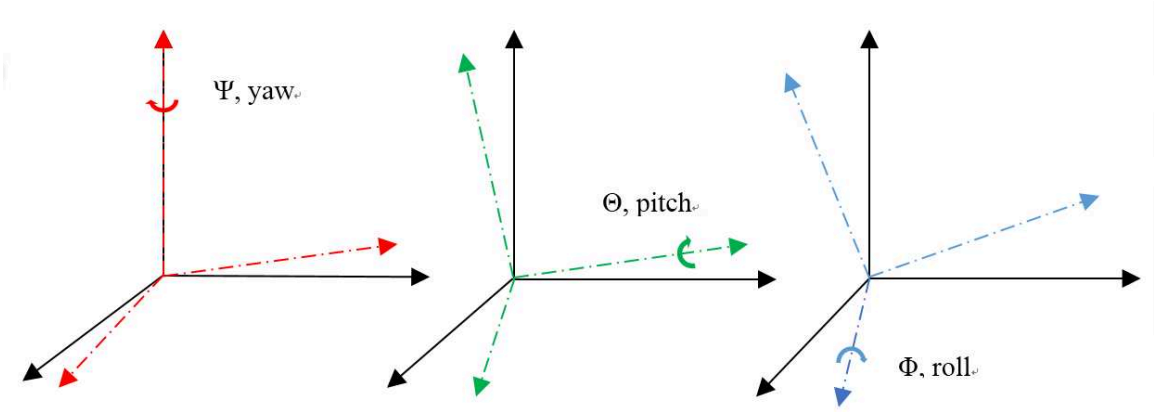

Figure 4.1: Schematic of the yaw-pitch-roll motion in terms of the Euler angles ψ, θ,ϕ[5]

$$Q = \begin{pmatrix} cos\theta cos\psi & sin\phi sin\theta cos\psi - cos\phi sin\psi & cos\phi sin\theta cos\psi + sin\phi sin\psi \\ cos\theta sin\phi & sin\phi sin\theta sin\psi + cos\phi cos\psi & cos\phi sin\theta sin\psi - sin\phi cos\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{pmatrix} \tag{4.2}$$

With this Q, the basis for a body frame could be transformed to the basis for a fixed spatial frame using equation:

$$e = Q^T E \tag{4.3}$$

Since 2-D images can only be affected by roll(ϕ) and pitch(θ), the angle around the z-axis is considered to be zero(ψ=0). Therefore, the equation for 2-D image processing is:

$$e = \begin{pmatrix} cos\theta & sin\phi sin\theta & cos\phi sin\theta \\ cos\theta sin\phi & cos\phi & -sin\phi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{pmatrix}^{T} E \tag{4.4}$$

Fig 4.2 and Fig 4.3 are the figures before and after pre-processing.

Figure 4.2: Original iceberg image

4.2 Contour Extraction

Contour extraction is a critical step for the algorithm, because it can directly influence the accuracy of the result. If one of the extracted contours is wrong, the final result can be completely meaningless. This is also the reason why a new algorithm needs to be developed for above-water iceberg 3-D modeling and volume estimation, since currently, almost all 3-D software is based on 2-D images that use SIFT to extract key information from images.

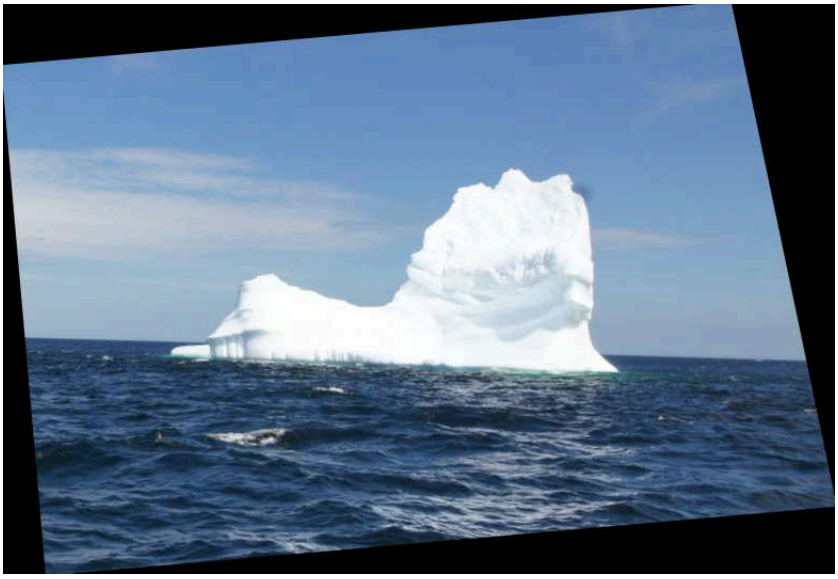

Figure 4.3: Iceberg image after pre-processing

However, SIFT is not sufficient to determine the exact contours of icebergs. I used two different strategies to extract the contours.

1) Region of interest. This is a manual method to extract the contour by using lines to select the iceberg region on an image in Matlab (Fig 4.4). After manual selection, the contour can be saved in a matrix. This method is not independent of manual work, but it is accurate and fast, therefore it is suitable for offline processing and simulation.

2) Automatic contour finding based on hierarchical image segmentation - see details in section 2.2 (Fig4.5) Only one layer from RGB color is used here to reduce the complexty of calculations. For automatic processing, an accurate algorithm which can process automatically is a must for this algorithm. The advantage of this method is that it can be processed without any manual work. However, it takes more time to process and discrepancies may still exist. Here the hierarchical image segmentation algorithm mentioned above are used.

The extracted contours are saved in separated files which is readable by Matlab for

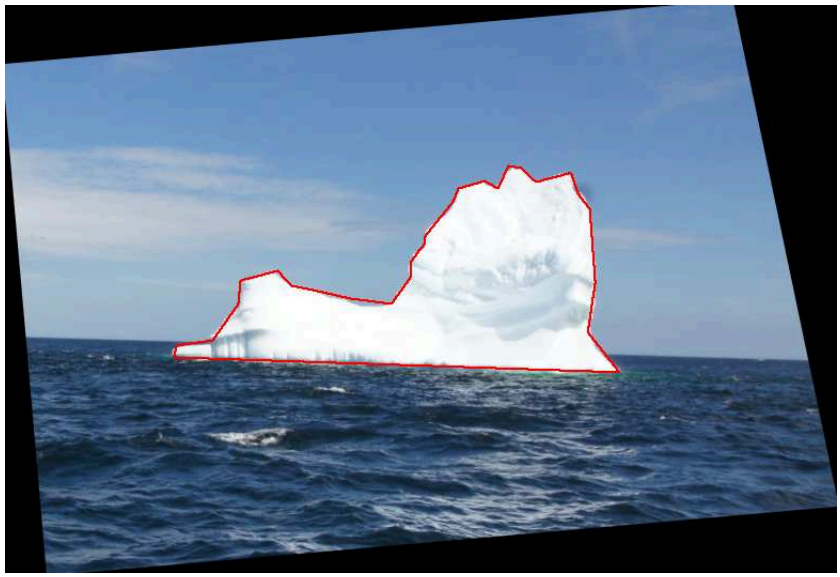

Figure 4.4: Manual contour extraction using manually selected region of interest

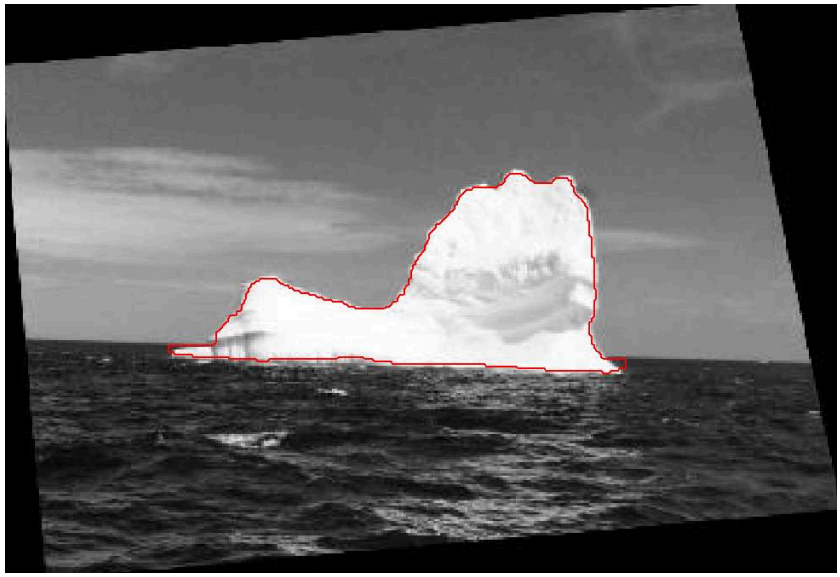

Figure 4.5: Automatic contour extraction using hierarchical image segmentation

future use.

4.3 Feature Matching

Once features have been identified and extracted from all of the images, they are then matched with each other [23]. This is known as the correspondence problem. In this thesis, the features being matched from the images are the contours extracted from 2-D iceberg images.

Although the algorithm used for contour finding is quite robust, it sometimes generates incorrect contours. For determining the volume of the above-water iceberg, it is preferable to dismiss the incorrect contour, otherwise it will result in a large error. Since the images are collected around the iceberg from different angles, two images taken in close proximity will have some similarities. Since the width of the iceberg can change significantly depending on the angle from which the picture was taken, the height will examine if the contour is correct. The contour is considered incorrect if the ratio of the height of the two contours are above a given threshold, in which case the match is then rejected. I analyzed the height of three contours that were captured from a similar angle, to figure out a standard height to use for comparison. If this fails, another height will be added for comparison until a standard is found. A nearest neighbor search strategy [23] is then used for the remaining contours until all incorrect contours are dismissed.

By deleting incorrect contours and using only the remaining correct contours, a 3-D model of a single iceberg can be reconstructed. However, the shape is meaningless without its parameters being calculated. The next section presents the method used in estimating the parameters using the collected data.

4.4 Image Post-Processing

Even though the contours of icebergs with their horizontal bottoms are obtained, the volume intersection can still not be performed. For each image, an iceberg's size (in pixels) on an image is different due to the significant changes of the focal length and the distance from the camera to the iceberg. Because the absolute height of the iceberg does not change, all the contours can be resized to have the same vertical pixels using Matlab or Octave, and the horizontal pixels will resize proportionally.

Before putting all the contours together in OpenSCAD, there are two important issues to address.

1) Contour placement in OpenSCAD (a programming solid 3-D modeler). If the contours are directly imported into OpenSCAD, they will be randomly placed in the 3-D space. The solution is to fix a peak point for those contours when importing into Octave. However, as the CAD files are printed by Octave in the first quartile in x-y coordinates, they will not retain the same point when rotating with the z-axis in OpenSCAD. To resolve this problem, a small piece of code written in FreeCAD can move the peak points to the base point of the 3-D coordinate. Therefore, the peaks will always be on the z-axis irrespective of how they are rotated.

2) Drifting angular velocity. I have explained how to estimate the drifting angular velocity, as well as how important it is to eliminate the influence of the rotation. In this step, this problem is to be solved. The heading towards the iceberg is measured by the ASC as absolute angle. As the iceberg drifts, the heading can not reveal the angular difference of the iceberg. Since building the 3-D shape of iceberg, the real angle of the viewpoints towards the iceberg is necessarily needed, otherwise significant errors will be caused. The

way to solve this problem is to address the estimated angular velocity of the iceberg, which equals the measured heading difference minus the real angle.

4.5 Shape Building

OpenSCAD [2] is a programming solid 3-D CAD modeller, which is not an interactive modeler, but rather a 3-D compiler based on a textual description language. The current geometric shape can be imported into OpenSCAD, and rotated in three dimensions. For the images collected on the ASC, rotating in 2 dimensions is enough. It could be expandable if more images are added from other viewpoints, e.g. from above using a quad-copter. After post-processing of the contours, the contours can be imported into OpenSCAD to place them at the appropriate viewpoints (Fig 4.6). Seen from this figure, the contours are placed at different viewpoint around the virtual iceberg. Then the contours are extruded linearly, and cut out the non-overlap areas. A 3-D model of the iceberg is generated.

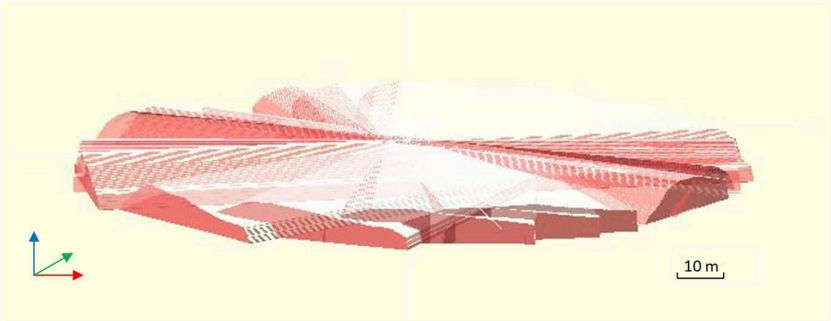

Figure 4.6: Volume intersection in OpenSCAD

The resolution of the iceberg model can be enhanced if I choose an appropriate number of intersections, which will be discussed in 5.3.

4.6 Automation Connector

Fig 4.7 is the flow chart of the whole process, and the part in a red square is the part involved in my software algorithm. To independently build the 3-D model without any manual processing, some code snippets (Appendix) are needed to connect the processing steps. Here I will show that there are specific parts that need to be connected.

1) Why use Octave? Although Matlab can extract the contour of the iceberg, it cannot generate a readable CAD file independently. Since Octave can generate a CAD file (.dxf) easily, Octave performs the function as a connector.

2) Why use FreeCAD? With the CAD file, it can then be imported into OpenSCAD for further processing. However, OpenSCAD requires the 2-D DXF files in "DXF R12", otherwise the files cannot be supported by OpenSACD. It is easy if the unsupported DXF is imported into other software, i.e. LibreCAD, and then exported. However, it will add some manual work into this process. Therefore, only software which can be controlled by programming can be used to tackle this problem. FreeCAD is written in Python and it comes with its own Python interpreter. Depending on the Python console, FreeCAD can be controlled by coding to transform the unsupported DXF format into "DXF R12" format.

3) Why use a shell script? This software algorithm is connected by a shell script. A shell script is designed to be run under Unix shell environment. Each line of the shell script can act as a fully normal Unix command, and it can provide a convenient variation of user defined processing procedures automatically. Using a shell script can also make it easy to transplant the processing procedures to a micro-controller, e.g. BBB. In this algorithm, shell script is like the wire which strings pearls (code from different software) together.

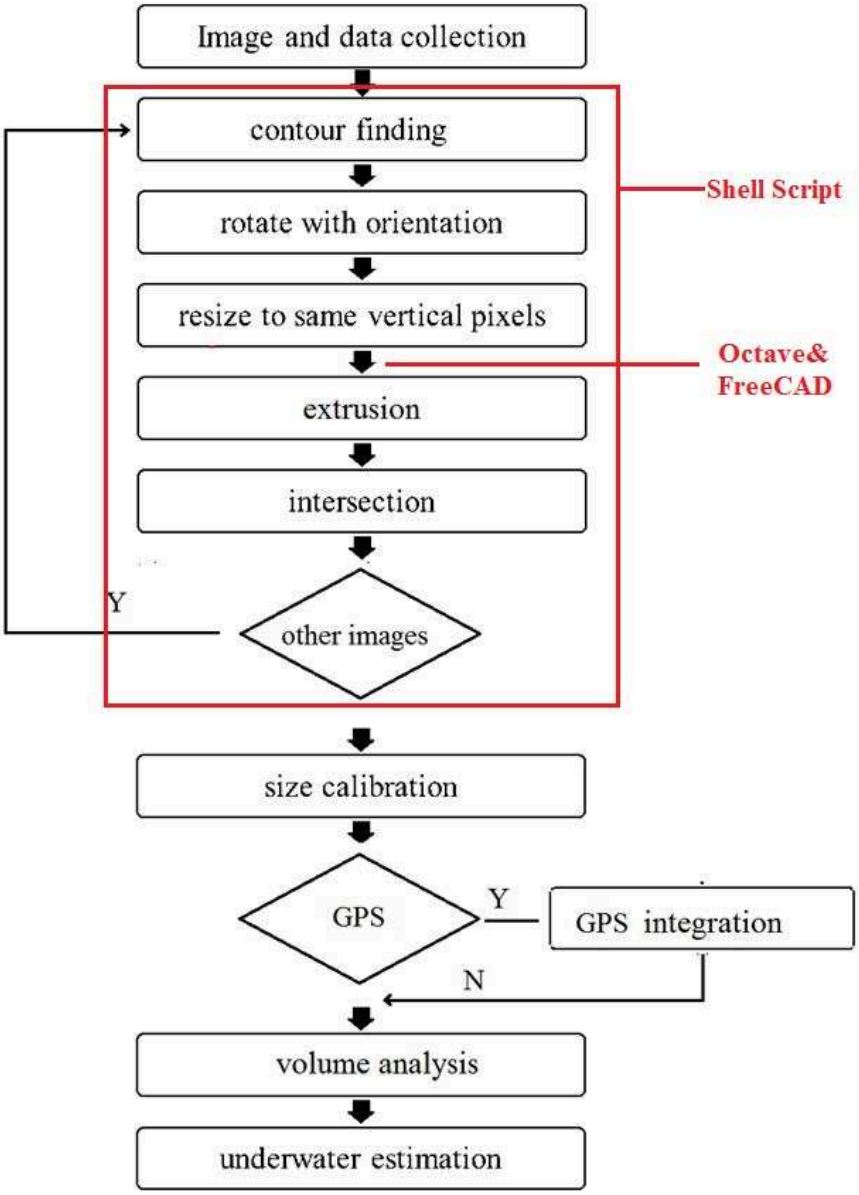

Figure 4.7: Flow chart of surface imaging method

4.7 Parameter Estimation

After having built the 3-D model, actually the 3-D visual hull, of the above-water iceberg, parameter estimation for this iceberg model is essential to quantitatively understand the iceberg.

1) GPS location. The first step is to plot the ASC route, then the points on the iceberg could be determined with respect to the heading of the vehicle, as well as the distance between the iceberg and the vehicle is measured. In this way, the visual hull of the iceberg can be related to the earth coordinates, and it interfaced with the underwater portion of the iceberg.

2) Size calibration. In chapter 2.4, I mentioned using height measurement to calculate the size of the visual hull. However, the height measurement is not possible for automatic collection of the ASC. Another way to calibrate the iceberg is to determine the size of the iceberg using LRF or LIDAR.

3) Volume estimation. The volume of the visual hull can be estimated using the disc integration method. The model is constructed from many polygons of different thickness. I choose a small thickness, then cutting the 3-D shape into layers. For each layer, the points inside the layer are already known, so I can combine these points to form a polygon. As the thickness of layers approaches 0, the sum of the layers is approaching to the volume of the iceberg. With the results of the 3-D shape, and the volume of the above water portion of the iceberg, I am able to estimate the underwater volume and an approximate keel depth of the iceberg derived from the empirical shape information.

4.8 Top-View Shape Integration

The issue of the visual hull problem, associated with the visual blind area, was addressed in Chapter 2.1. Along with the camera viewing parameters, the silhouette defines a back-projected generalized cone (visual hull) that contains the actual object. The visual hull phenomenon will affect the resolution of the size of the volume of the above-water iceberg when the top-view of the iceberg is an obvious concave. It is therefore very helpful to obtain a top-view shape of the above-water portion iceberg.

For the grounded iceberg, the top-view shape could be generated using equations from 2.3.1. There is also a complicated condition that the icebergs are drifting. For those icebergs, the drifting velocity can be estimated as stated in 2.3.2. Another solution here is to use a quad-copter to capture iceberg images above the iceberg, showing the top-view of the iceberg directly and accurately. The top-view shape could be easily added to the procedure of shape building in OpenSCAD by rotating to the appropriate angle.

As the top-view shape is obtained, it is necessary to classify the icebergs into two groups, the normal iceberg and the iceberg with significant concave from the above(Fig 2.2 and Fig 2.3). The classification can make it easy to analyze the resolution and thus make better estimation for the real size of the above-water iceberg. For most cases, the top-view shapes are not necessary to be added to the visual hull of the iceberg if there is no significant concave form to the iceberg was observed from the above.

Here, where possible, I will use both (LRF and UAV) as it allows me to determine the strengths and the limitations of the two techniques. If the top-view shape is really required, it does not necessarily need to be collected from an aerial vehicle, such as quad-copter. It could be obtained, for example, by using a one-dimensional LRF whose methodology is

addressed in 2.3.1 and 2.3.2, since the LRF on ASC is used to generate the shape of the iceberg close to the waterline, which could be considered as an approximate top-view shape under most circumstances.

Chapter 5

Resolution Estimation

After executing the algorithm, the volume of the visual hull of an iceberg can be measured. Several error sources can be associated with this algorithm. First and foremost, every iceberg has a different size ratio of its visual hull over that of its real shape. Second, errors could arise from the data collection instruments. For example, there are errors in the GPS sensor and the AHRS sensor. Third, I estimate the drifting velocity of the iceberg using a linear model, but the iceberg could drift in a more complicated way. Because the total data collection time could be reduced to a very short time, and the sensors are of good quality and have small errors, the most significant error source likely arises from the visual hull, which means the model is always a convex shape seen from the top-view, while it actually can be concave. In this chapter, I use the PERD Shape Database to analyze the error of the visual hull and give an estimated range of accuracy, by assuming the data to be the standard iceberg shape and creating a 3-D iceberg visual hull using the method developed in this thesis.

5.1 PERD Iceberg Sighting Database

The PERD Iceberg Sighting Database has been updated annually and improved since 1998 [29]. The database provides the valuable data for iceberg research. It plays an important role in planning future activities, and possibly as input to other initiatives such as decision-making, and iceberg management planning aid technologies [31]. I uses the Iceberg Shape Database [30], because there is detailed 3-D information for iceberg sails and keels, coming from a number of sources, including National Energy Board, Mobil, C-CORE, Coretec Inc, Bercha and Associates (Alberta) Ltd., Petro-Canada, and the Bedford Institute of Oceanography. Most of the data comes from the Grand Banks and Labrador region. Detailed iceberg keel data and sail data from this database are in the form of x, y, z coordinates for different depths and heights relative to the sea surface, allowing reviews of the details for each iceberg. Since data can be searched, sorted, printed, and exported in various formats, it is suitable for detailed iceberg's simulation to analyze the resolution of the algorithm in this thesis.

In this thesis, I focus more on small and medium size icebergs with height of around 10-60 meters, which are normally seen around the Newfoundland coast and threaten the offshore structures and ships. Huge icebergs, such as ice islands, are not considered in this thesis.

5.2 Simulation

The thesis makes use of the Iceberg Shape Database with data from 1999 and detailed sail data in x, y, z coordinates. By using the detailed sail data, the 3-D shape of the above-

water iceberg can be built directly. This shape can be regarded as the ”real” shape and the ”real” volume of it can be calculated. Screen shots at different heading were taken of this ”real” shape and these screen shots were used as 2-D images to build a 3-D model (visual hull) of this iceberg. I took screen shots using the same strategy - size depending - as in the field work, that is, took more screen shots for larger icebergs and less screen shots for smaller icebergs. I used these screen shots to perform volume intersection algorithm. Each screen shot works as one intersection. The simulated volume of the visual hull can also be calculated when the new x, y, z data are got. By comparing significant number of the ”real” and simulated volume of the icebergs, the resolution can be estimated.

The iceberg data used in this chapter are r11l02, r11l03, r26f1i21, r26f2i06, r26f3i03, r26f3i05. The ”real” shape and simulated shape can be compared in the following figures.

Table 5.1: Comparison on icebergs from PERD Database

Iceberg Number	Database Shape	Surface Imaging Shape
r11l02	Fig 5.1	Fig 5.2
r11l03	Fig 5.3	Fig 5.4
r26f1i21	Fig 5.5	Fig 5.6
r26f2i06	Fig 5.7	Fig 5.8
r26f3i03	Fig 5.9	Fig 5.10
r26f3i05	Fig 5.11	Fig 5.12

As seen from the side-views of these models, they are similar to the ”real” shapes. However, the visual hull shortcomings are shown clearly in the isometric views or top views of these icebergs, which will significantly enlarge the size of the models. These above-water icebergs don’t have an obvious concave form, so we can put these icebergs in a group. The icebergs with an obvious non-convex shape will be analyzed separately.

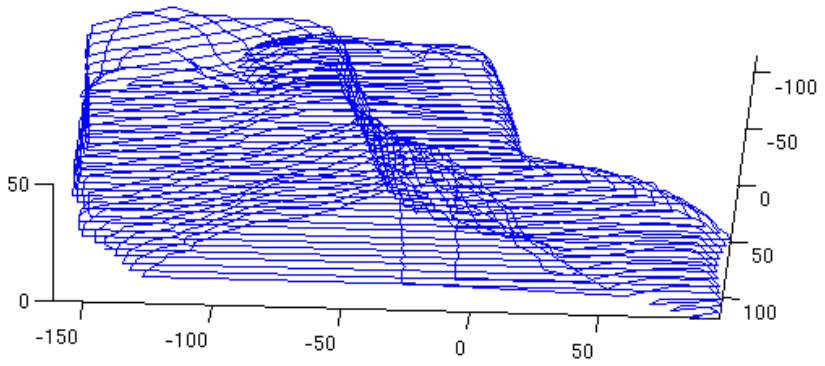

Figure 5.1: R11l02 - shape from the PERD Database

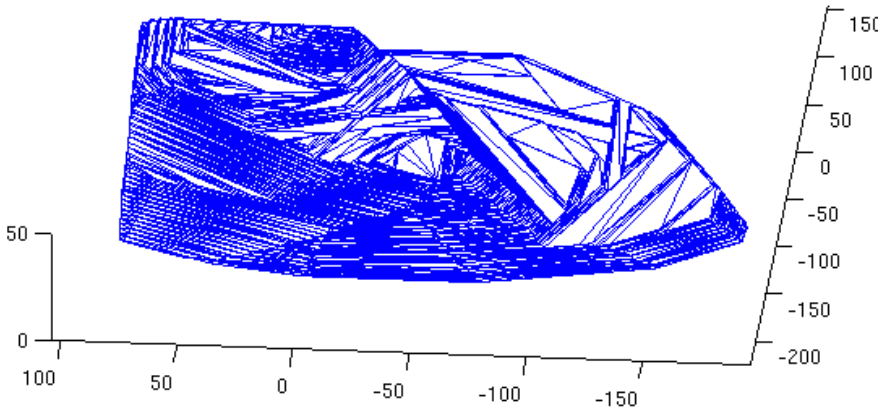

Figure 5.2: R11l02 - 3-D model using method in this thesis

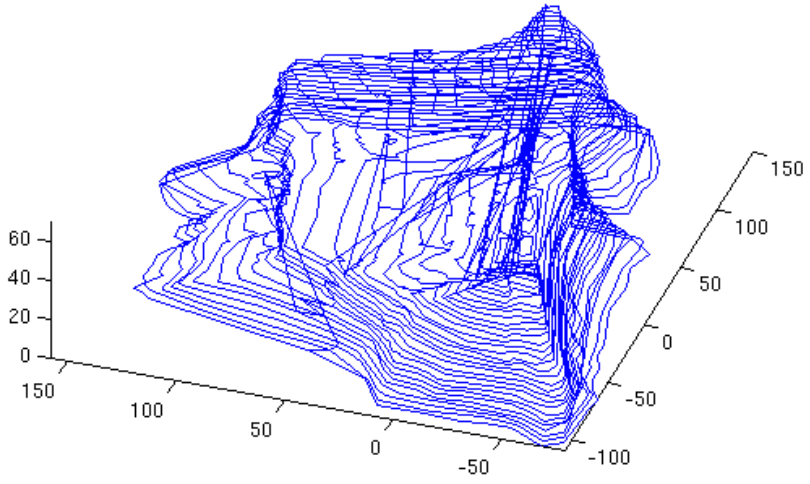

Figure 5.3: R11l03 - shape from the PERD Database

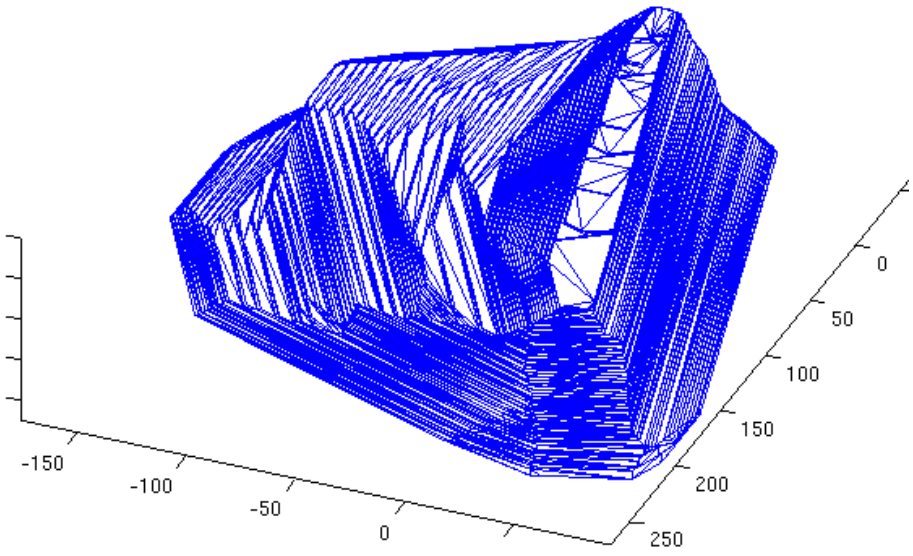

Figure 5.4: R11l03 - 3-D model using method in this thesis

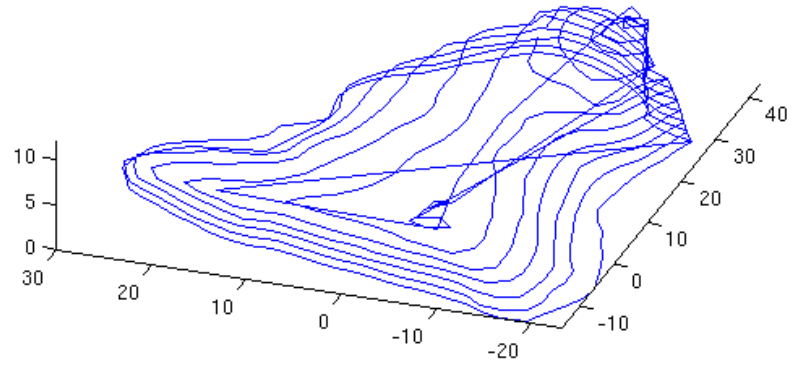

Figure 5.5: R26f1i21 - shape from the PERD Database

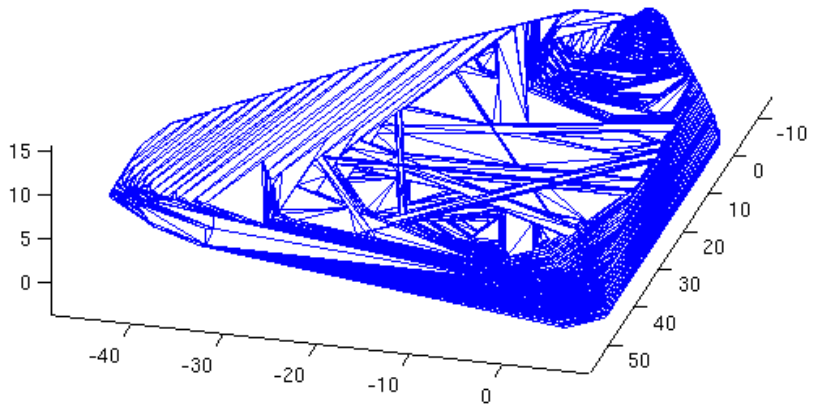

Figure 5.6: R26f1i21 - 3-D model using method in this thesis

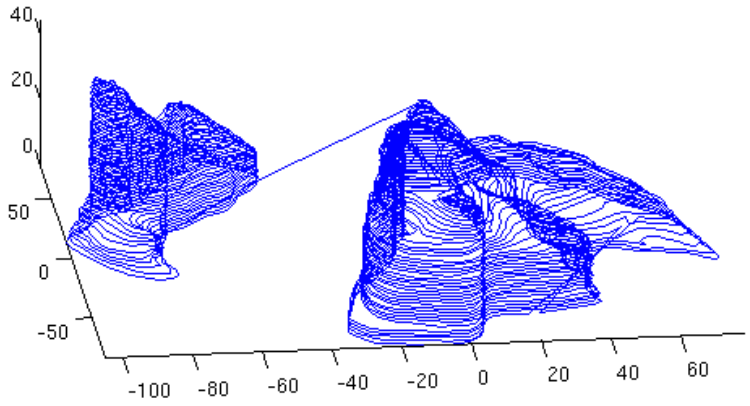

Figure 5.7: R26f2i06 - shape from the PERD Database

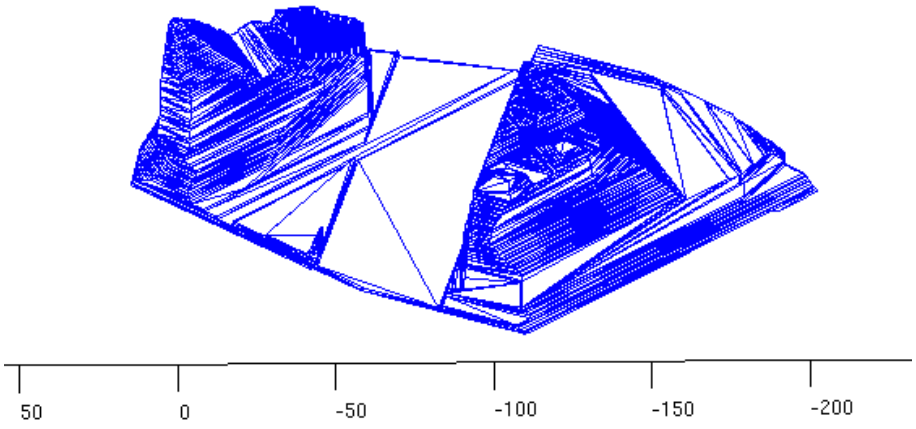

Figure 5.8: R26f2i06 - 3-D model using method in this thesis

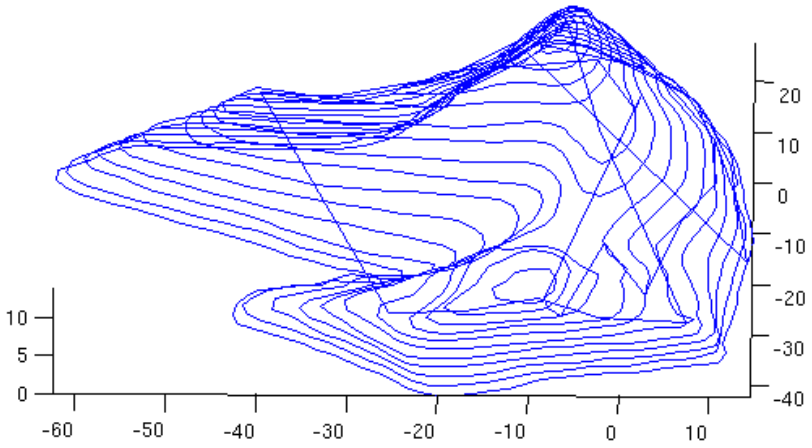

Figure 5.9: R26f3i03 - shape from the PERD Database

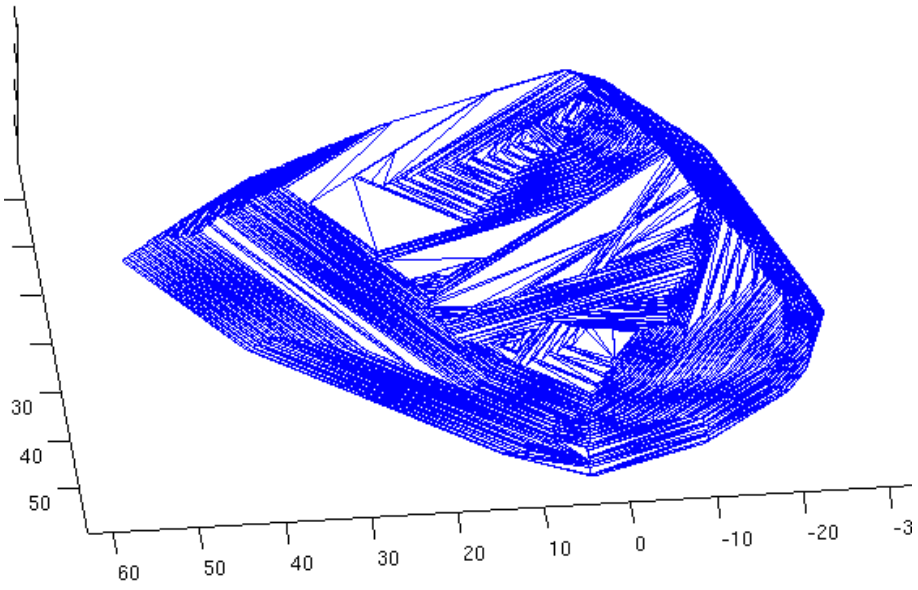

Figure 5.10: R26f3i03 - 3-D model using method in this thesis

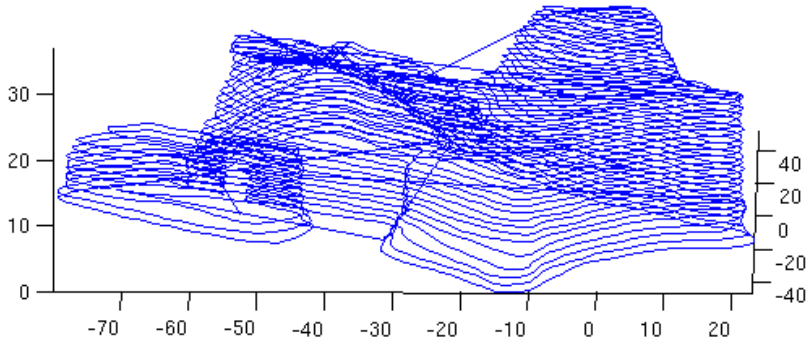

Figure 5.11: R26f3i05 - shape from the PERD Database

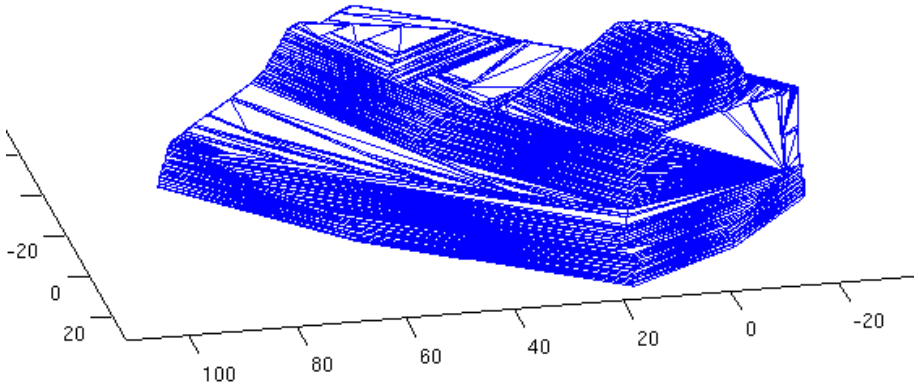

Figure 5.12: R26f3i05 - 3-D model using method in this thesis

5.3 Analysis A

With the known database, the volume of the ”real” shape can be calculated. The questions to be analyzed here are: (1) with how many intersections can we get a reliable model; and (2) what is the resolution of the reliable model.

The six icebergs presented above are used here to perform this error analysis. I performed the volume intersection on different numbers of side-views, and calculated the volumes. To show clearly, the ratio of the volume of the models over the ”real” volume is plotted versus the numbers of intersections. (Fig 5.13)

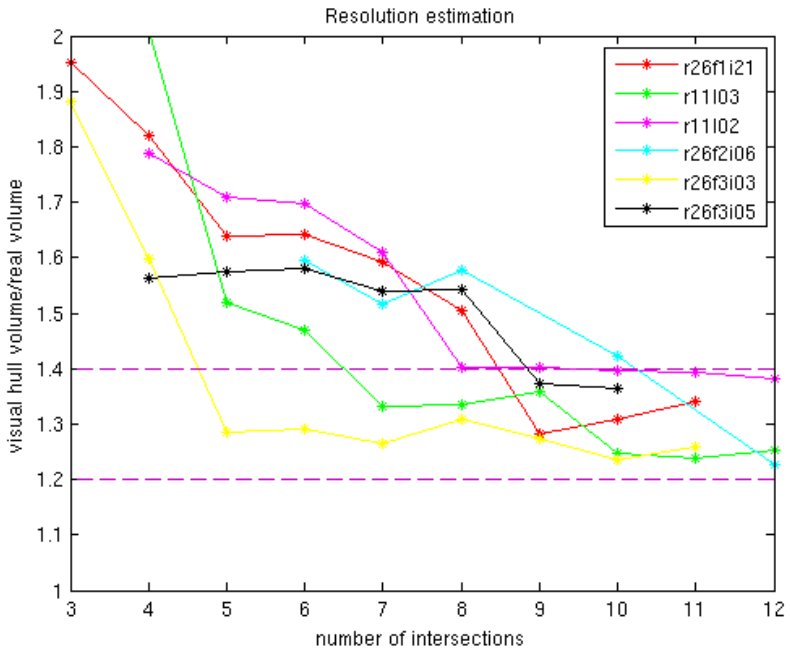

Figure 5.13: Resolution estimation on the six iceberg models with different intersections

As shown in Fig 5.13, the result tends to converge after 10 intersections, no matter the size of the icebergs. That is to say, a reliable model could be obtained with more than 10

intersections. With more than 9 intersections, the resolution continues to improve, but not significantly. In the process of selecting intersections for the simulation, I avoid selecting the same ones, for example, at the angles of 0 and 180, 10 and 190. This suggests that 20 or more images should be taken during field works to avoid repetition of contours and make sure that enough images have been taken to perform this algorithm. Also at least 20 images are needed to be taken even for small icebergs.

With more than 9 intersections, the results become stable and the estimate is between 120%-140% of the real value. It is interesting to note that there is, as expected, a reasonable bias since the visual hull problem tends to over predict the volume, which is theoretically due to the characteristic of the visual hull. In order to correct the result to produce a resolution of 90%-110%, a correction parameter (130%) needs to be added here. Typically I use the calculated volume over 130%, thus a volume within 10% error can be obtained. Fig 5.16 shows the estimated resolution after this correction, and the resolution is within 10%.

5.4 Analysis B

Few icebergs have an exact convex top-view. I analyzed the icebergs with normal top-view in Analysis A in order to get a simple probability frame. However, in only rare cases were the top-view shapes of the iceberg significantly concave, for which icebergs the correction parameter (130%) is not sufficient to correct their volumes' resolution within 10%. In such a situation, I need to do extra processing to get a reliable result.

As for the definition of significantly concave, there is no documentation on this topic. I analyzed the top-view shapes of the iceberg sails in PERD Shape Database, and listed some

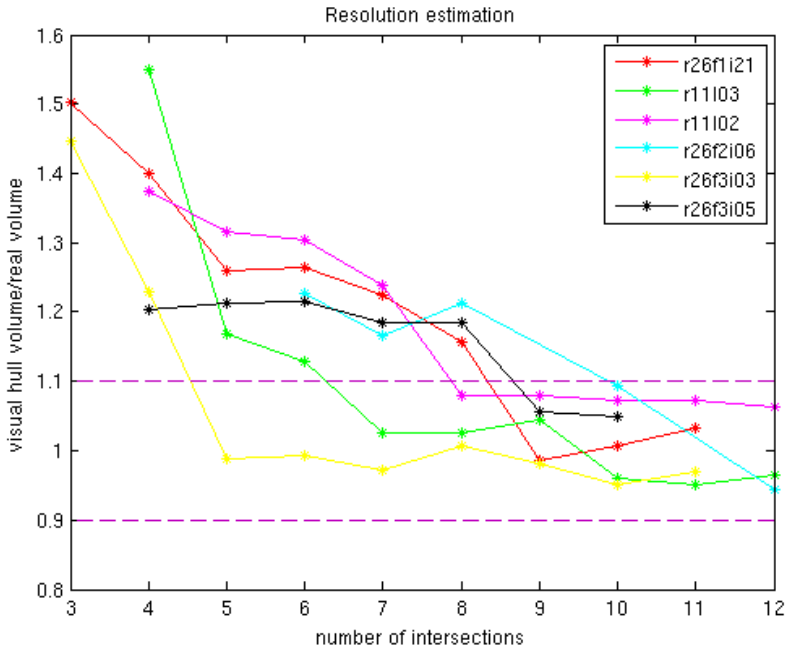

Figure 5.14: Estimated resolution on the six iceberg models using the correction parameter

shapes in Fig 5.16. These top-view shapes can be considered to be insignificantly concave.

The iceberg in Fig 5.16 is r11l01 from the PERD Shape Database (1999). As seen in this figure, the concave portion is almost a quarter of the whole shape. The visual hull influence could not be neglected for this iceberg, since it is an obvious outlier compared to those analyzed above (Fig 5.17).

In this case, I can add the top-view shape generated by the LRF to carve the unnecessary part of the visual hull. Fig 5.18 is the real model of r11l01, and Fig 5.19 is the model of it. The real volume is 1.93e+5 m^3, while the model's volume is 1.76e+5 m^3. The estimated iceberg size is 91.18% of the "real" size, which is a significantly optimized result compared with the visual hull for this kind of iceberg.

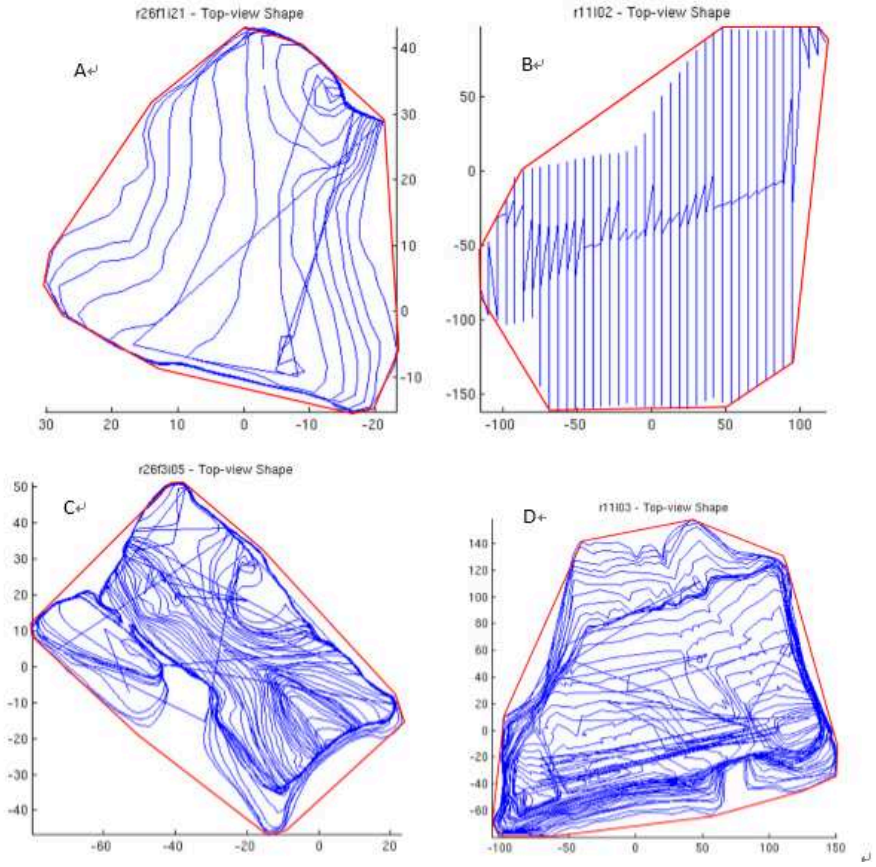

Figure 5.15: Some top-view shapes of icebergs

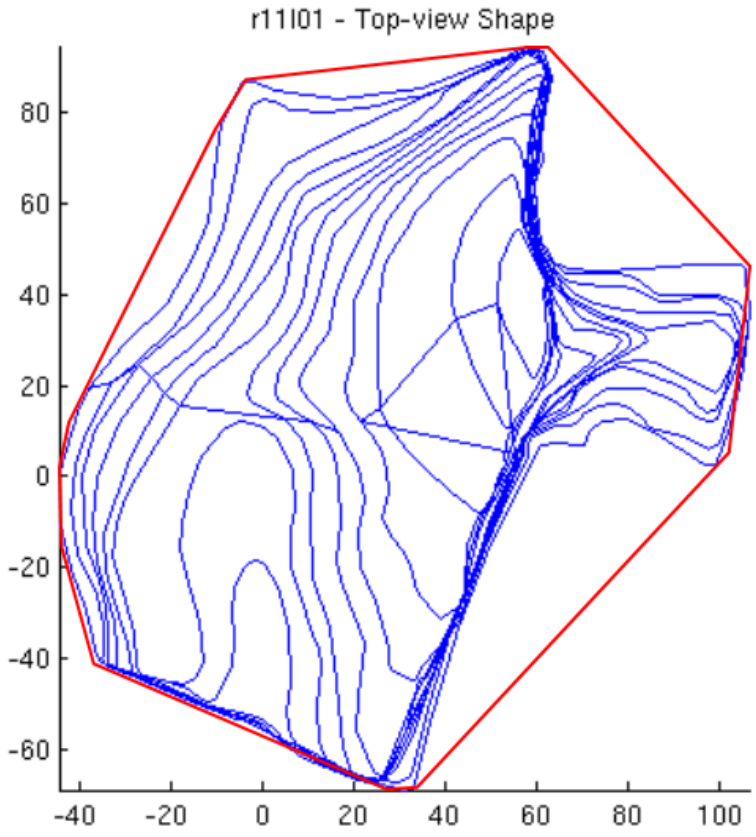

Figure 5.16: Top-view shape of r11l01

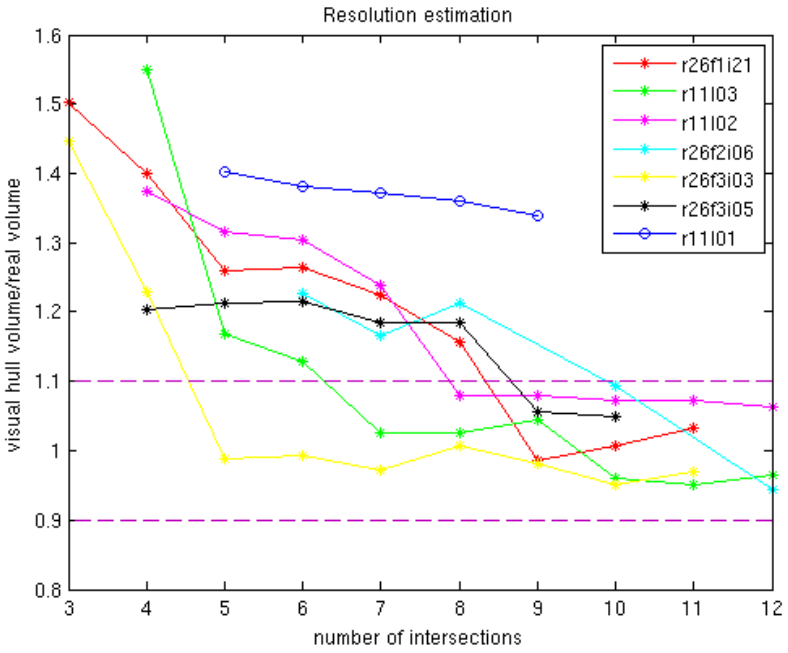

Figure 5.17: R11l01 is an outlier on the analysis

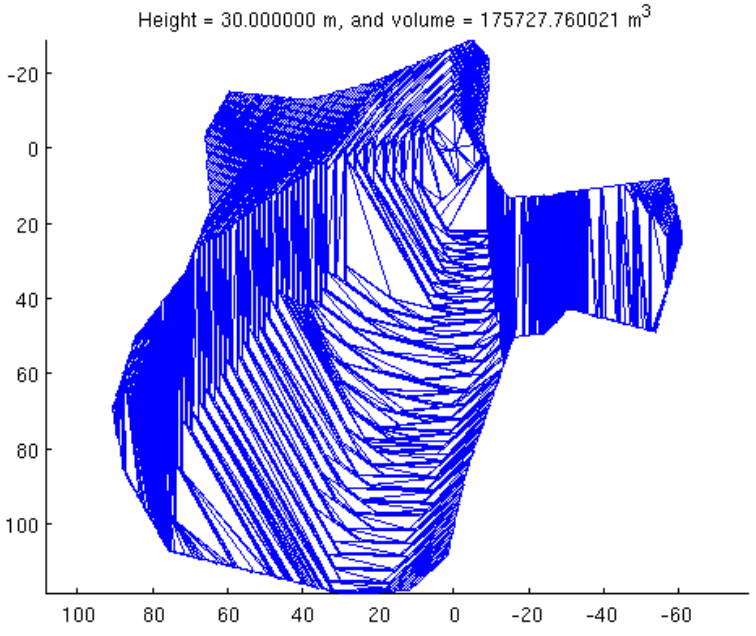

Figure 5.18: R11l01 - 3-D Model using method in this thesis

Chapter 6

Results and Discussions

6.1 results

In 2014, three field trails were performed, and six icebergs were observed. I will present their orignal images as well as their 3-D models in this chapter.

Table 6.1 shows the icebergs collected during the iceberg season of 2014.

Table 6.1: Summary of Icebergs

Iceberg	Day	Nearest Harbour	Images	Model Projections	3-D Model
No. 1	May 26,2014	Holyrood	Fig 6.1	Fig 6.2	Fig D.1
No. 2	June 18,2014	CBS	Fig 6.3	Fig 6.4	Fig D.2
No. 3	July 13, 2014	Twiilingate	Fig 6.5	Fig 6.6	Fig D.3
No. 4	July 13, 2014	Twiilingate	Fig 6.7	Fig 6.8	Fig D.4
No. 5	July 13,2014	Twillingate	Fig 6.9	Fig 6.10	Fig D.5
No. 6	July 13, 2014	Twiilingate	Fig 6.11	Fig 6.12	Fig D.6

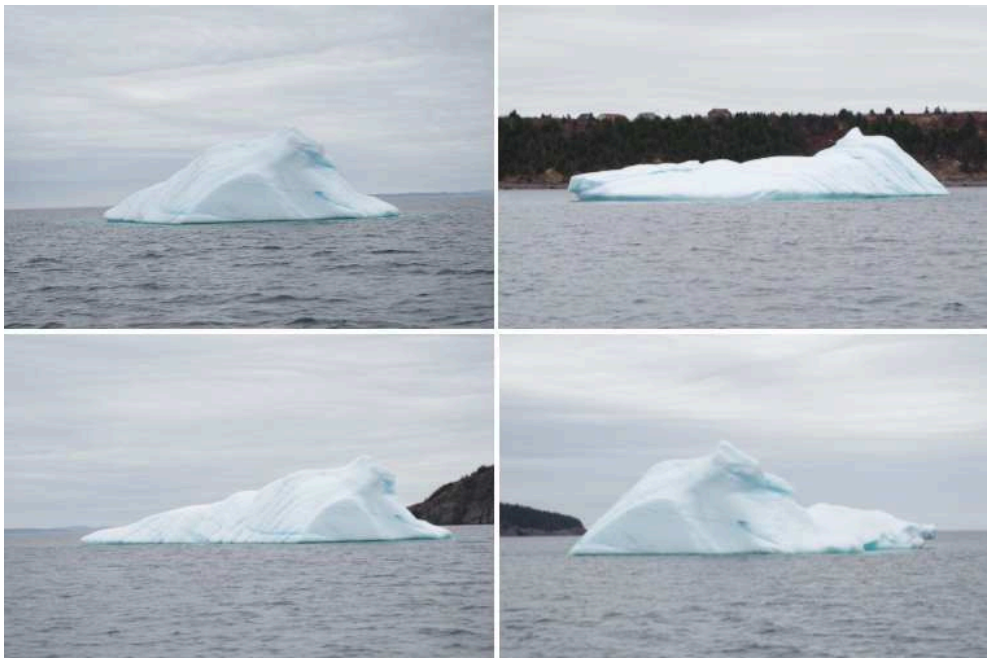

Figure 6.1: Iceberg No. 1 Images

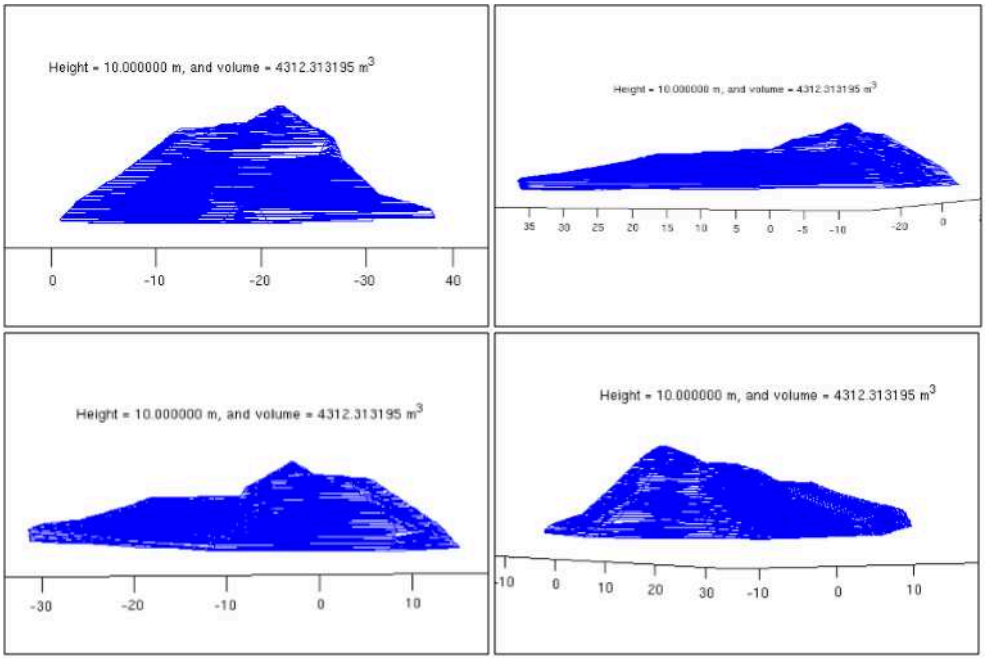

Figure 6.2: Iceberg No. 1 Model projections

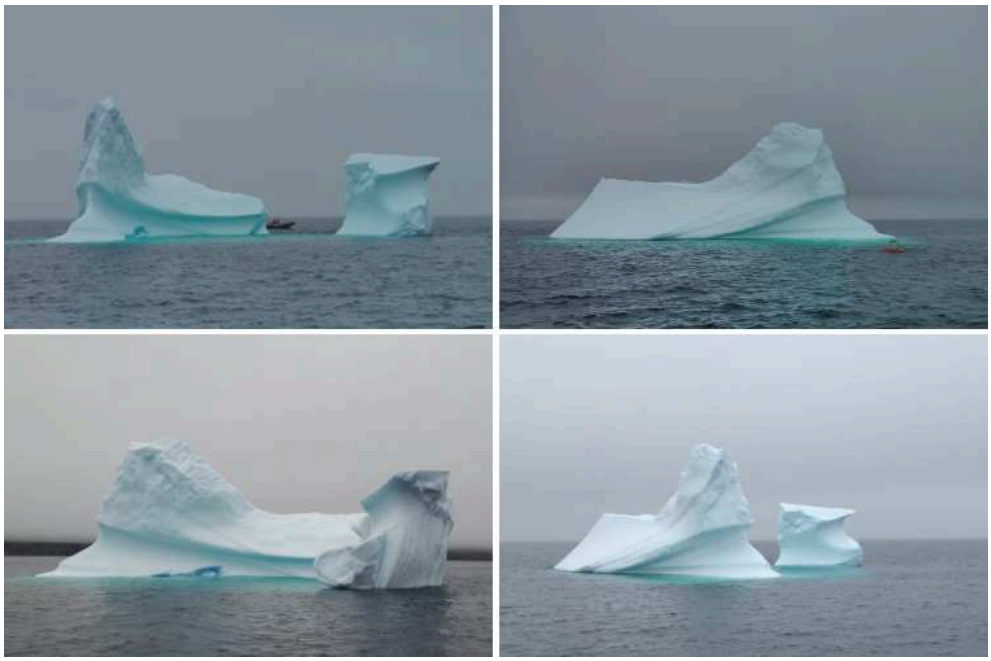

Figure 6.3: Iceberg No. 2 Images

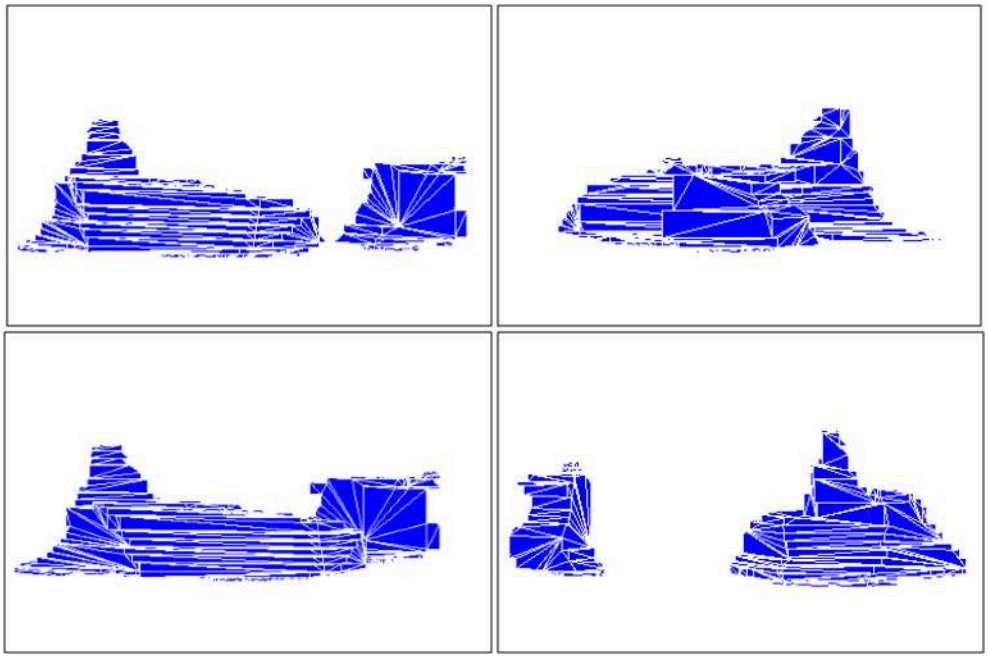

Figure 6.4: Iceberg No. 2 Model projections

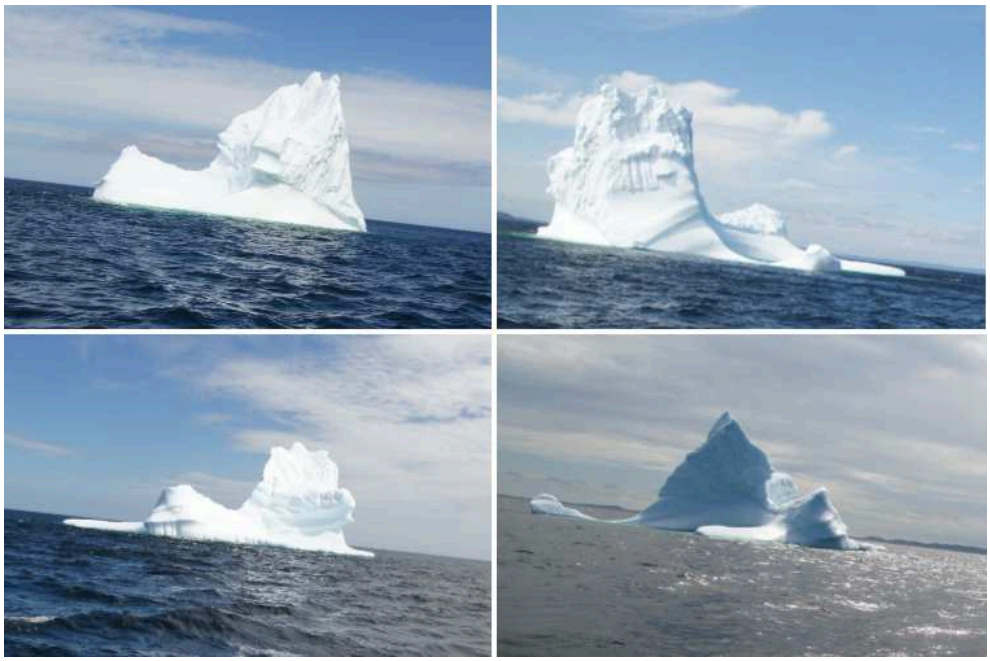

Figure 6.5: Iceberg No. 3 Images

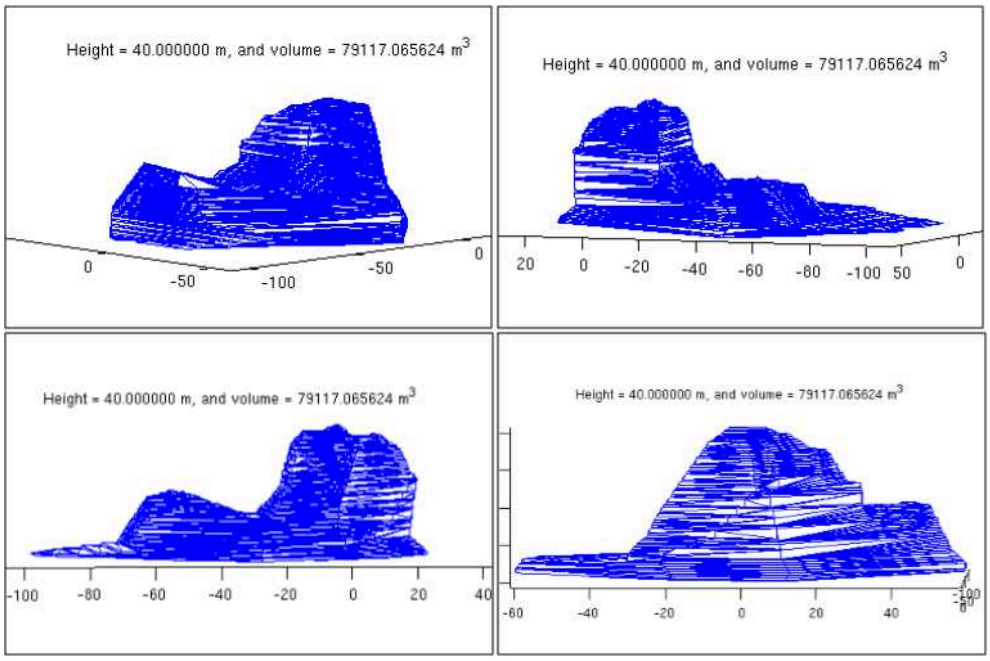

Figure 6.6: Iceberg No. 3 Model projections

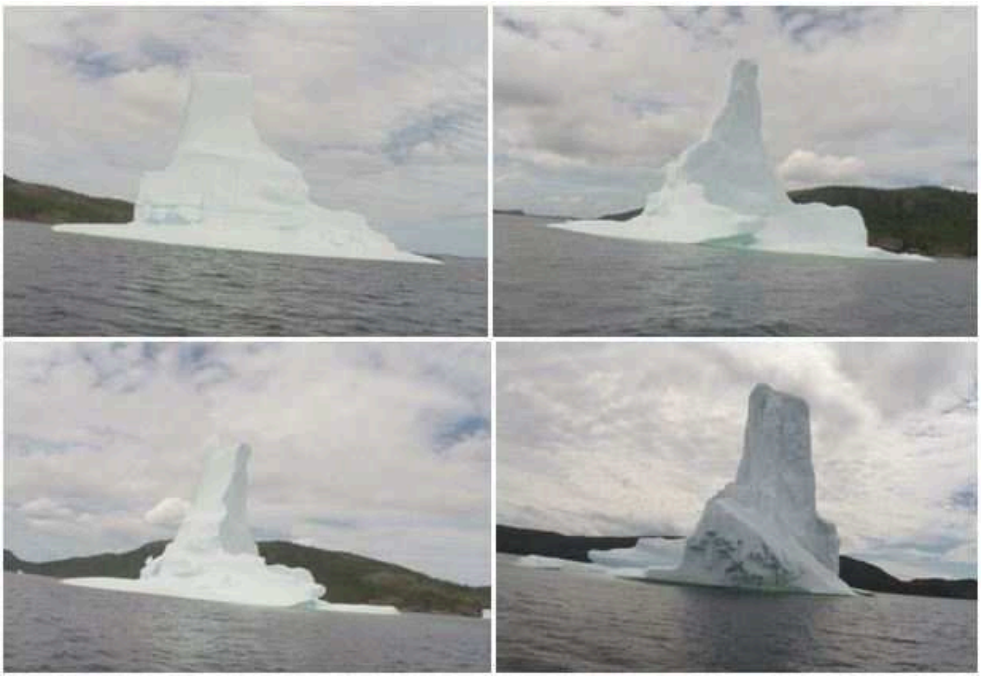

Figure 6.7: Iceberg No. 4 Images

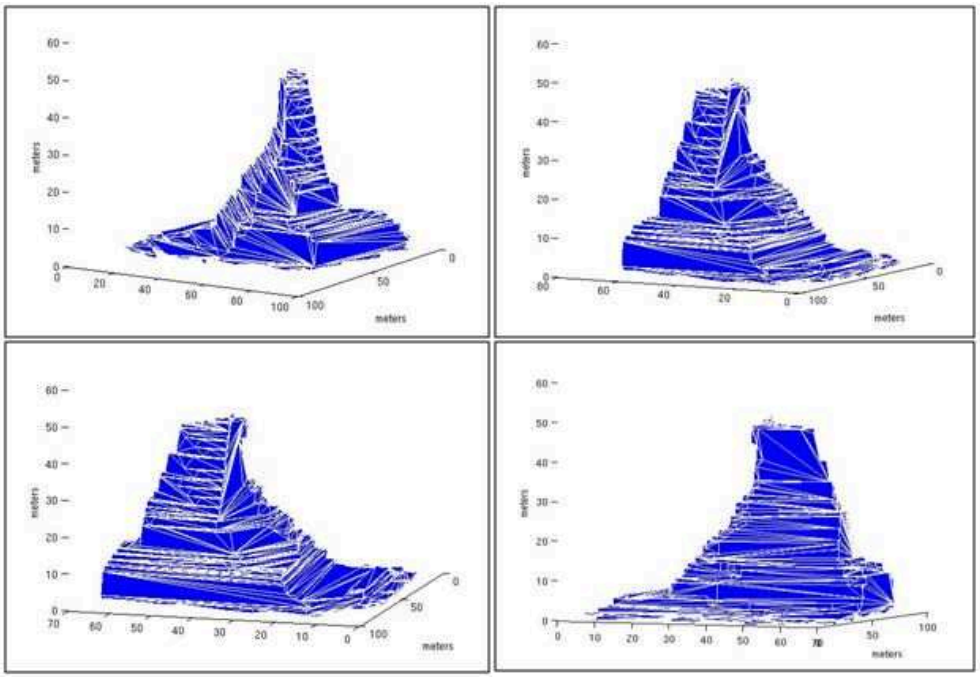

Figure 6.8: Iceberg No. 4 Model projections

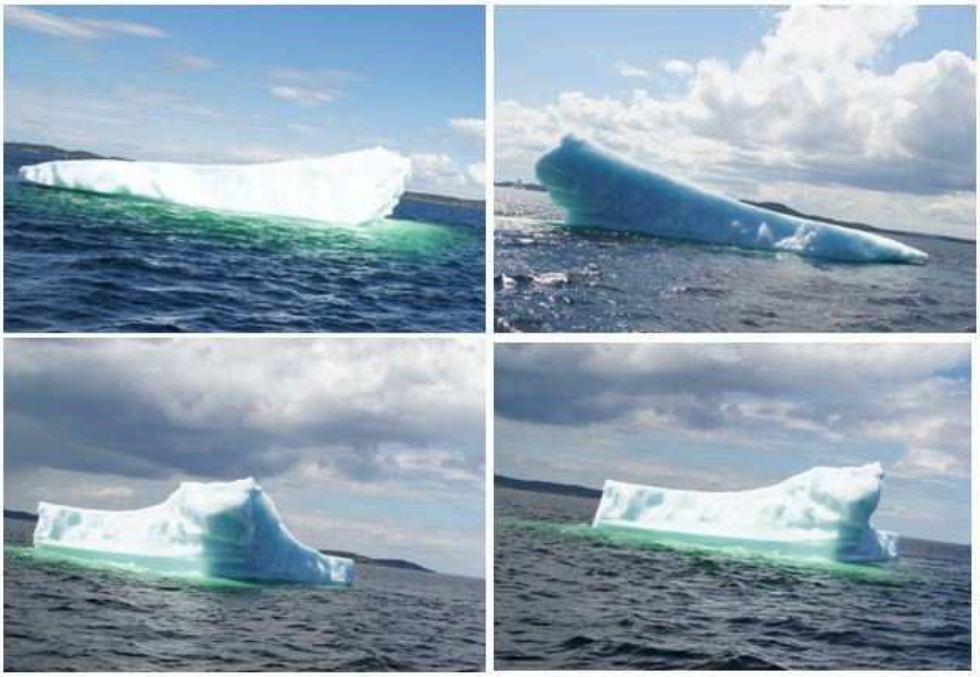

Figure 6.9: Iceberg No. 5 Images

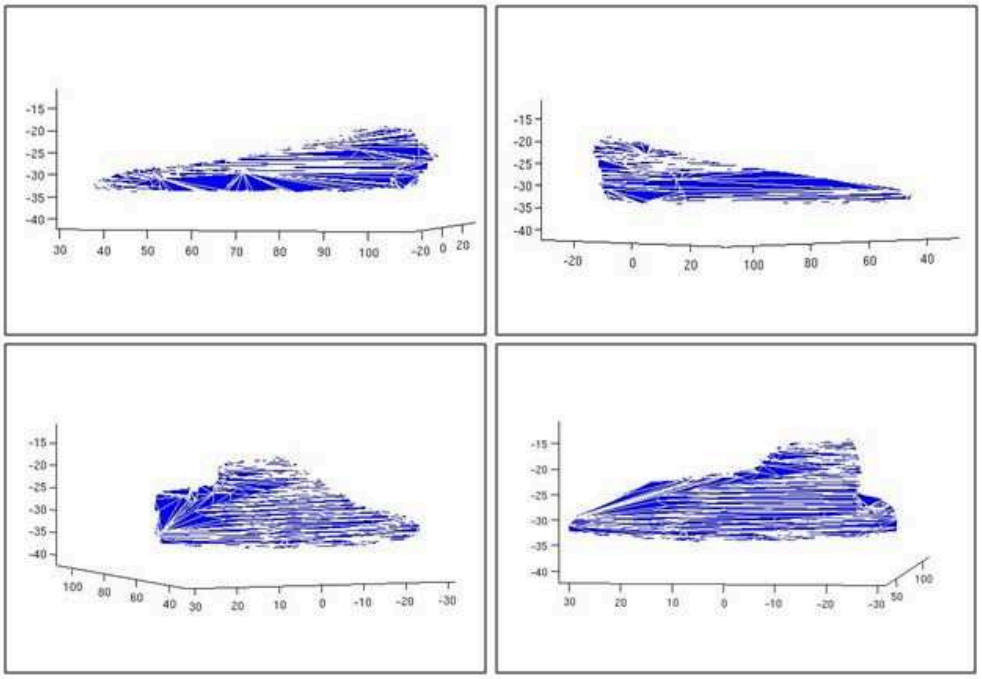

Figure 6.10: Iceberg No. 5 Model projections

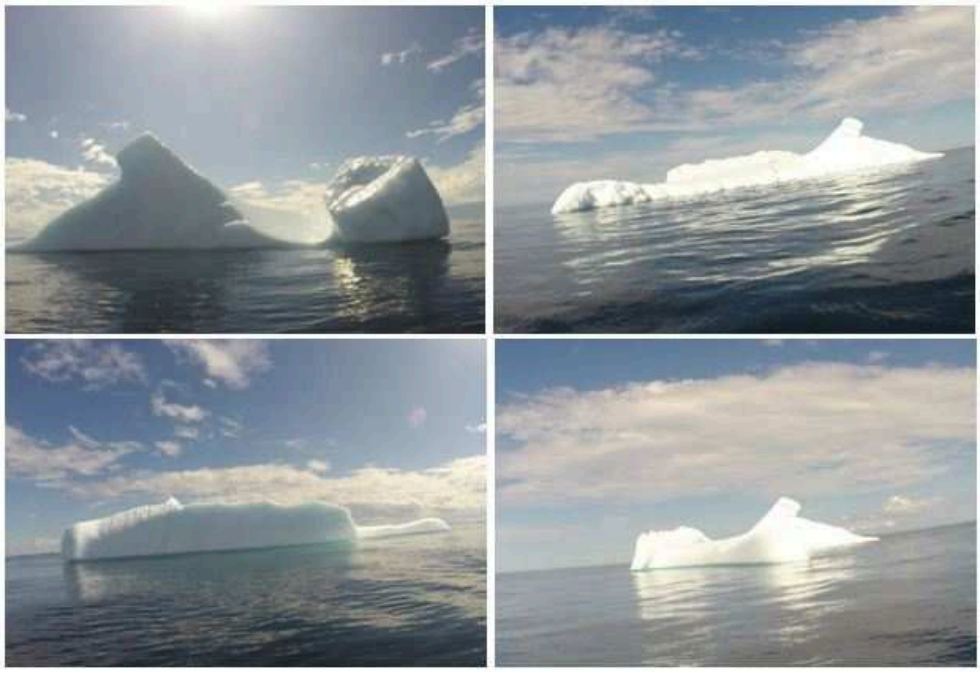

Figure 6.11: Iceberg No. 6 Images

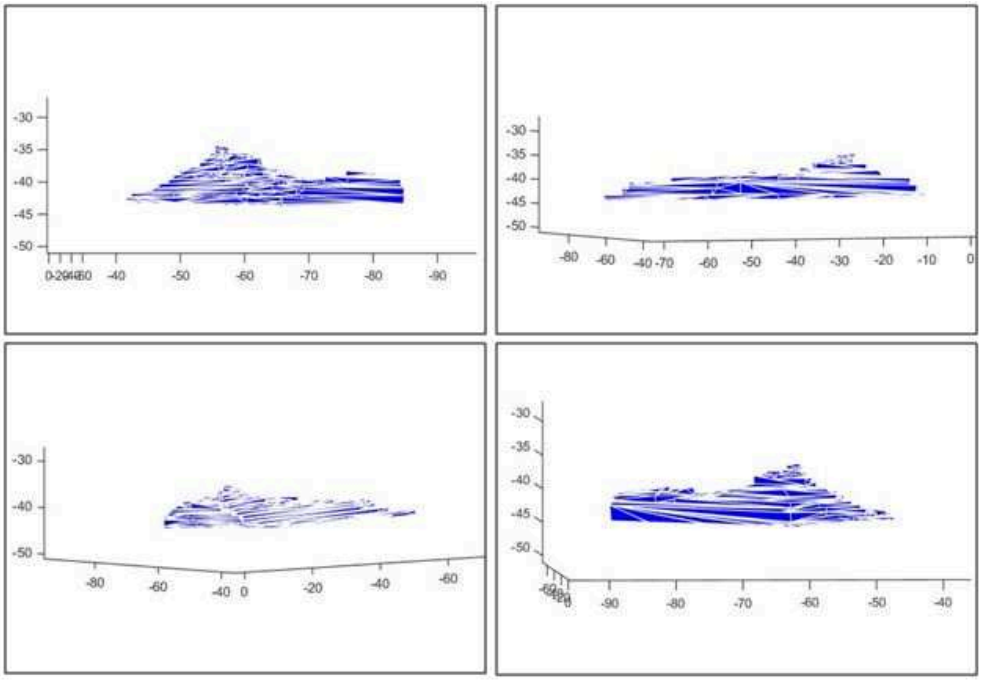

Figure 6.12: Iceberg No. 6 Model projections

6.2 Discussion

Iceberg No. 2 is a dry-dock shaped iceberg. Because of its particular shape, the volume cannot be calculated using method described previously. In order to estimate the volume of the dry-dock iceberg, the model needed to be separated into two parts that are calculated separately. The sum of the volume of these two parts is the volume of the dry-dock iceberg with $\pm 10\%$ errors, according to the simulations run using the PERD Shape Database in Chapter 5.

In addition to the surface imaging, the results are also verified partially with a laser range finder (Fig 2.10). Since two size calibration (height measurement and laser range finder) are applied, I can verify each other by the comparison between the two results. By plotting them in the same figure (Fig 6.7), it can be demonstrated that the 3-D model and the laser range finder top-view shape are to the same length scale, which gives us more confidence that the results are reasonable. The top-view shaped can also be used to correct the visual hull shortcomings.

The surface imaging method is a feasible way to profile the above-water iceberg. Generally, it takes 10 minutes to build an iceberg model for the data from the PERD database, and it will take longer to process the more complicated images collected in the field. The exact time also depends on the size of the iceberg, the light, sea state, and the weather conditions. However, the results do not have high resolution because of the visual hull limitation. Also, the resulted 3-D model built by surface imaging method is not geo-referenced, which means it is not registered with GPS position automatically, which gives difficulty in the combination with the underwater portion iceberg profiled by sonar.

Chapter 7

Iceberg Profiling Using LIDAR

7.1 Introduction

LIDAR (Light Detection and Ranging) is a remote sensing method that uses light in the form of a pulsed laser to measure ranges (variable distances) to a stationary point[16]. These light pulses combined with other data recorded by IMU and GPS generate precise, three-dimensional information about the shape of an object and its surface characteristics. I used the ASC to carry a LIDAR device in order to scan the surface of icebergs so that their 3-D point clouds can be obtained.

The reason why I deploy LIDAR is because the surface imaging system has the big drawback associated with the visual hull, and LIDAR can produce a dense point cloud to correct the 3-D shape of icebergs. On the other hand, although the LIDAR system can produce 3-D point clouds, extra calculations which can bring in error are needed when transforming the point clouds into solid models. Therefore, neither methods is perfect but each is useful for profiling of the above-water icebergs. The point cloud generated from the

LIDAR can be easily matched up with the sonar data since they are both geo-referenced. In practice, the LIDAR and sonar point clouds can be aligned first, and transform the whole iceberg point cloud into a solid model.

7.2 System Design

7.2.1 LIDAR sensor

The Velodyne VLP-16 sensor (Fig 7.1) is small, light, 3-D distance and calibrated, and it can perform 360 degree measurement in real-time. The Velodynes LiDAR Puck supports 16 channels, 300,000 points/sec, a 360 degree horizontal field of view and a 30 degrees vertical field of view, with ± 15 degrees up and down. The Velodyne LiDAR Puck does not have visible rotating parts, making it highly resilient in challenging environments.

Figure 7.1: Velodyne VLP-16 LiDAR Puck

7.2.2 Electrical Component

The main components of the LIDAR electrical control box are micro controller - Beaglebone Black (BBB), LIDAR User Interface Box and switch. (Fig. 6.2)

7.2.2.1 Micro Controller

The same micro controller of the surface imaging system is used for the LIDAR. The micro Controller is aimed to record the live stream from the LIDAR. To record the live stream, Wireshark is installed in the Beaglebone Black (BBB). A shell command (tshark -w /stroage host 192.168.0.201) was written in BBB and auto-started when powered on.

7.2.2.2 User Interface Box

The User Interface Box comes with the VLP-16 sensor because there is no internal polarity nor over voltage production in the sensor. Therefore it is imperative that the Interface Box and/or protective circuitry is incorporated in every installation. I took the sensor along with the User Interface Box into my custom LIDAR system.

The User Interface Box communicates with the micro controller through ethernet port. I used a switch in order for micro-controller to communicate with the LIDAR system and camera system. The power supply for the whole LIDAR system was on the ASC.

7.2.2.3 GPS Unit

The sensors can synchronize their data with precise GPS supplied time. Synchronizing to the GPS pulse-per-second (PPS) signal provides us the ability to compute the precise time of each observation point. A Garmin GPS-18-LVC was configured to issue a one-

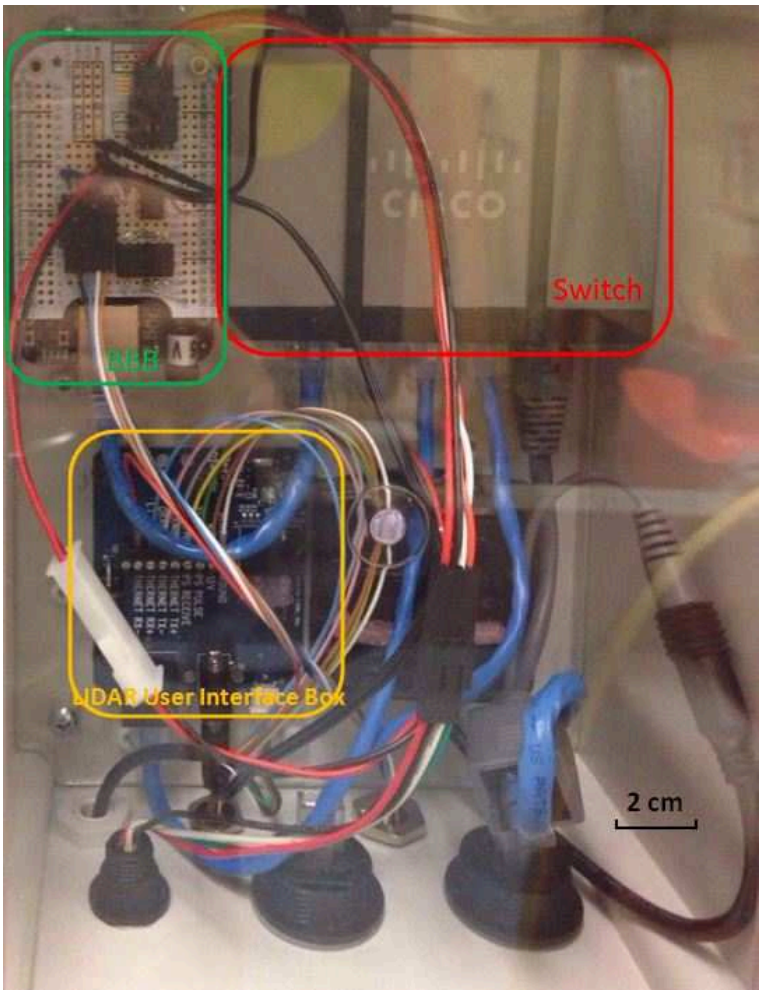

Figure 7.2: LIDAR Electrical Control Box

per-second synchronization pulse in conjunction with a once-per-second NMEA $GPRMC message, so that the User Interface Box can receive and interpret it.

7.2.2.4 Inertial Measurement Unit

The VLP-16 sensor does not have a compatible Inertial Measurement Unit. In order to measure the orientation of the sensor, which is important when it is on the sea surface, the movement of orientation is significant. Since the LIDAR system has a GPS Unit to synchronize with GPS time, and the ASC also has a precise GPS Unit as well as an AHRS (Attitude and Heading Reference System) - The Microstrain 3DM-GX3-25 AHRS sensor, I can match up their time stamps and synchronize the UTM positions and orientations. The ASC will communicate with the the LIDAR system through the CAN bus, and the micro-controller will record the CAN bus messages to match the CAN bus messages logged by the ASC captain node.

7.3 Data Processing

7.3.1 Orientation of the sensor

In order to build a point cloud for one object, the sensor in space is needed to move to scan all parts of the scene. Because of the limitation of the range and the angle (Fig 7.3.A), the whole iceberg (especially not the top of the iceberg) beyond a certain range can not be observed if the iceberg is big. Therefore I changed the orientation of the sensor slightly (Fig 7.3.B), which means that extra effort is required in the post-processing part to change the point cloud back to the real spatial coordinates.

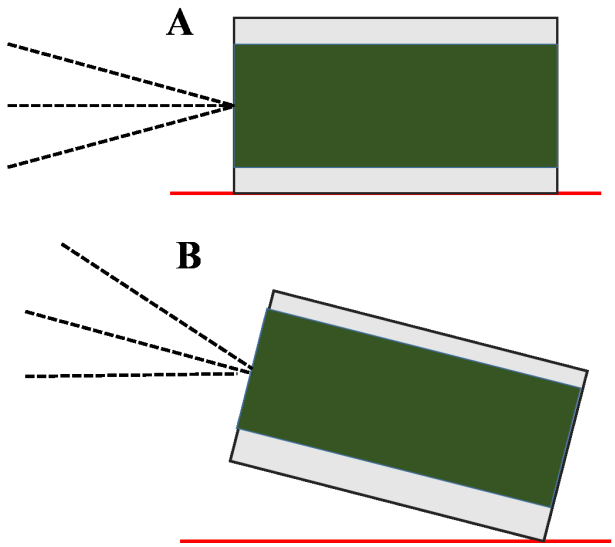

Figure 7.3: A: The Velodyne LIDAR scans about the horizontal axis from +15° to -15°; B: An alternative sensor orientation for the LIDAR on the ASC with scanning beam from 0° to 30°

7.3.2 Data Playback

The sensor's data is recorded in a .pcap file through Wireshark and stored in the SD card of the BBB. Velodyne LIDAR provides a pre-calibrated viewer for VLP-16 - Veloview. After deleting the checksum in the pre-recorded .pcap file, the file can be imported into Veloview directly. The data can be exported as XYZ data in CSV format. The exported files are sorted as scanning "frames". The sensor's scanning rate is 10 frames per second. Each frame contains around 30,000 points.

7.3.3 Geo-referencing

The raw frame data are needed to be accumulated to build a dense 3-D point cloud for each iceberg, as each scan only has limited points. Because the sensor takes data relative

to its own position, the sensor's orientation and position must be accounted at the same time. By knowing the position and orientation of the sensor at each moment, the LIDAR can be transferred to a single reference frame mathematically to generate a point cloud model. The process of translating the LIDAR data into a single reference frame is called geo-referencing.

7.3.4 Data Cleaning

After performing the algorithm for lidar data, the point clouds comprise the target, the ASC, the ship as well as other noisy reflection, such as mountains or birds. However, there is no way to automatically delete all noise points, since different noise for different targets can be involved. Therefore, I manually deleted the obvious outliers using CloudCompare, an open source software for point clouds post processing.

7.4 Result Analysis

7.4.1 Small Floating Iceberg

On June 10, the AOSL group did the first field work of 2015. There were few icebergs in Conception Bay during that period, and due to the technique problems of other systems, I only scanned one piece of floating ice using the LIDAR system. Since the target was floating, it was hard to get the whole scan of the iceberg because the frames are registered with GPS. The scanning rate for LIDAR is 10 frames per second. In Fig. 7.4 and Fig. 7.5, I added up 100 frames (around 10 seconds) to show the iceberg from one viewpoint. The point cloud is very dense.

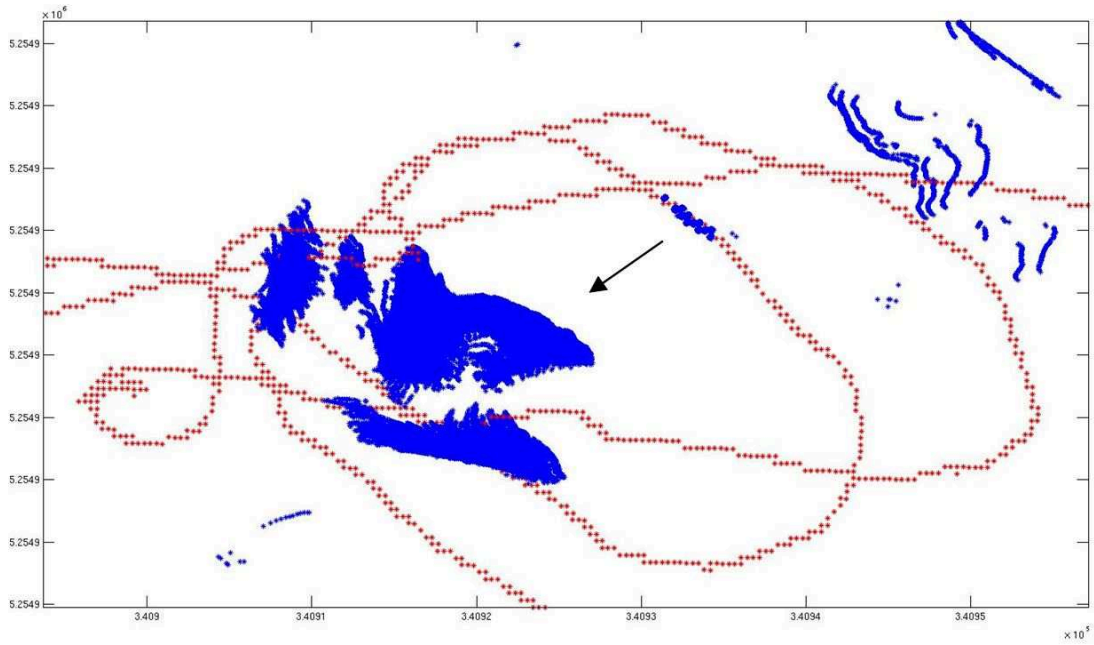

Figure 7.4: Floating iceberg from one viewpoint. The red dots are the GPS position of the ASC, the blue points are points detected by the LIDAR, and the black arrows shows the viewpoints towards the measured iceberg.

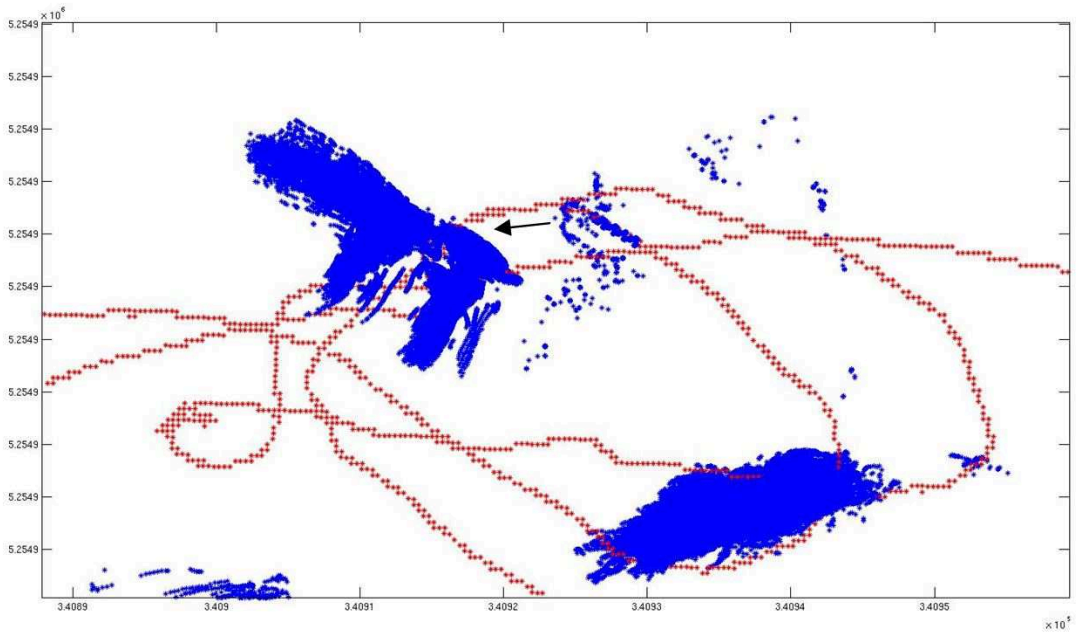

Figure 7.5: Floating iceberg from another viewpoint

7.4.2 Harbour Main Island

On July 27, 2015, the group went for another field trail to profile the Harbor Main Island (Fig 7.6) because there was no ice available for sampling. The LIDAR is mounted on the side of the ASC with 15 degrees orientation on x-axis (Fig 7.3.B) in order to have a 30° scanning angle above the waterline.

After data processing, Fig 7.7 shows the raw point cloud of Harbor Main Island and the GPS route of the ASC. Fig 7.8 shows the clean point cloud after manually cleaning using CloudCompare. These results are verified by inserting the GPS and AHRS information into the LIDAR package, which produces the same outcome as the algorithm mentioned above.

The advantage of using LIDAR is that LIDAR can create more accurate and dense

Figure 7.6: Harbor Main Island from Quadcopter

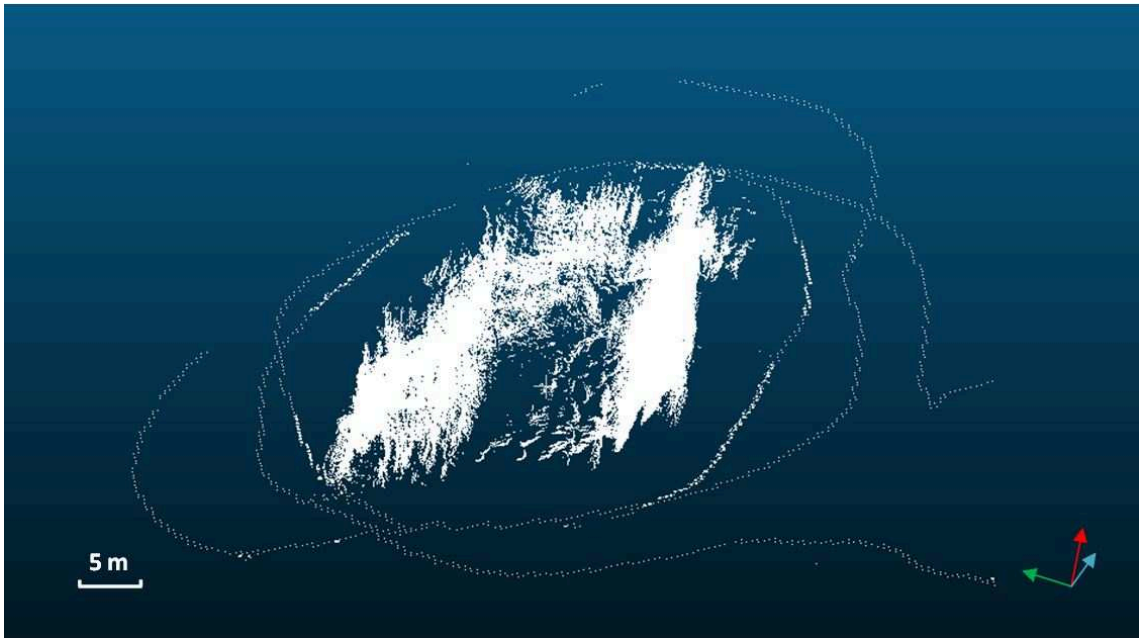

Figure 7.7: Raw point cloud from LIDAR and GPS route of the ASC

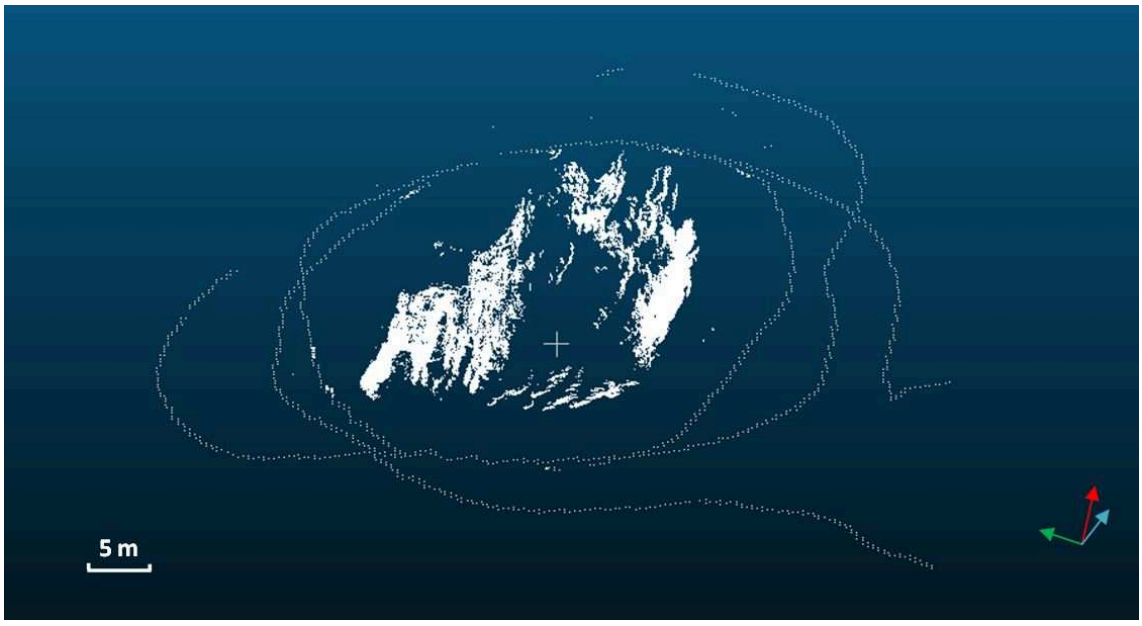

Figure 7.8: Clean point cloud of harbour main island

point clouds compared with the surface imaging method. The point cloud generated from the LIDAR can be easily matched up with the underwater portion of the iceberg since both sets of the data are geo-referenced. However, manual efforts are needed to clean the data and get a well rendered mesh. The error sources of the LIDAR are birds on the island, the boats around it and the accuracy of the LIDAR sensor itself and AHRS sensor.

7.5 Surface Imaging Method on Island

I also performed the surface imaging method on this island to produce the 3-D shape. In performing the surface imaging algorithm on Harbor Main Island, it is not fully automatic since the target has almost the same color as the background, which makes it extremely difficult to extract the contour automatically from the images. Also, the island is relatively flat but also corrugated, which makes the 3-D model low-resolution, non-manifold, and cannot be exported to calculate the volume.

Chapter 8

Conclusions and Future Work

The surface imaging method is economic, convenient, and offers fast processing compared with traditional profiling methods. The working principle has been established using surface optical images taken from different directions around the iceberg, co-registered with location, orientation of the camera as well as the sampling platform. The method could generate a rendered 3-D solid model of the above-water portion of the iceberg. The method is stable and robust since it is based on the fundamental SFS algorithm. Although the resolution is limited due to the visual hull problem, the results are still valuable for quick decision-making. However, this method does have some significant drawbacks. First, it requires a high-speed CPU to process images on board. Second, the rendered 3-D above-water iceberg model is difficult to register with GPS automatically making it challenging to integrate the above-water 3-D model with the underwater iceberg portion. Third, the concavity issue is addressed but cannot be resolved automatically on board. Given the above limitations, a LIDAR system is used to profile icebergs in order to overcome these obstacles. The LIDAR, which can measure the concave of the iceberg directly, can gener-

ate point clouds registered with GPS so that those data can be aligned with the sonar data for the iceberg. In future, the point clouds can be transformed to solid models. The data from the other sensor (e.g. the camera) can be used to fill the gaps, when there are data gaps in one sensor (e.g. LIDAR). Even when there are enough observations, the additional information can also be used to verify the validity of the overall solid models.

The 3-D model of the above-water can be used to estimate the volume of the underwater part of iceberg, which can also help to build up a reasonable assumption for the underwater iceberg shape if there is partial sonar data but not enough collected due to whatever reasons.

Currently, the surface imaging system and LIDAR systems have been built and integrated with the ASC in the AOSL. These were used to collect some valuable data in the field in the summer of 2015. At present, they can be operated only semi-autonomously. This means that the data can be collected autonomously, but manual work is needed in the post-processing. In the near future, the fully autonomous system with a powerful on-board processor will be developed. Another algorithm which could transform the point clouds to 3-D solid models is also required.

Bibliography

[1] *www.beagleboard.org*.

[2] *www.openscad.org/*.

[3] J. Aloimonos. Visual shape computation. *Proceedings of the IEEE*, 76(8):899–916, 1988.

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898–916, 2011.

[5] H. A. Ardakani and T. Bridges. Review of the 3-2-1 euler angles: a yaw–pitch–roll sequence. *Department of Mathematics, University of Surrey, Guildford GU2 7XH UK*, 2010.

[6] A. Barker, I. Skabova, and G. Timco. Iceberg visualization database. *PERD/CHC Report*, pages 20–44, 1999.

[7] L. Bo, X. Ren, and D. Fox. Kernel descriptors for visual recognition. In *Advances in Neural Information Processing Systems*, pages 244–252, 2010.

[8] R. Brown, P. Dunderdale, and J. Mills. On the development of an iceberg management planning aid. *Port and Ocean Engineering under Arctic Conditions POAC03 (Norway)*, 2003.

[9] C-CORE. Stability and drift of icebergs under tow. *1st Annual Atlantic Petroleum R & D Forum Presentation*, 2007.

[10] C.-H. Chien and J. Aggarwal. Model construction and shape recognition from occluding contours. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 11(4):372–389, 1989.

[11] G. Comfort. Investigation of techniques for continuously profiling ice features. *PERD/CHC Report*, pages 1–49, 1998.

[12] G. Crocker, B. Wright, S. Thistle, and S. Bruneau. An assessment of current iceberg management capabilities. *Contract Report for National Research Council Canada, Prepared by C-CORE and B. Wright and Associates Ltd., C-CORE Publications*, pages 98–C26, 1998.

[13] R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[14] Fugro. Integrated 3d iceberg mapping. 2014.

[15] T. Jebara, A. Azarbayejani, and A. Pentland. 3d structure from 2d motion. *Signal Processing Magazine, IEEE*, 16(3):66–84, 1999.

[16] D. KILLINGER. Lidar (light detection and ranging). *Laser Spectroscopy for Sensing: Fundamentals, Techniques and Applications*, 292, 2014.

[17] Y. Kim and J. K. Aggarwal. Rectangular parallelepiped coding: A volumetric representation of three-dimensional objects. *Robotics and Automation, IEEE Journal of*, 2(3):127–134, 1986.

[18] Z. Li and R. Bachmayer. The development of a robust autonomous surface craft for deployment in harsh ocean environment. *Oceans-San Diego*, pages 1–7, 2013.

[19] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *Computer Vision–ECCV 2008*, pages 43–56. Springer, 2008.

[20] M. R. Maire. *Contour detection and image segmentation*. PhD thesis, University of California, Barkery, 2009.

[21] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):530–549, 2004.

[22] W. N. Martin and J. K. Aggarwal. Volumetric descriptions of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1983.

[23] H. M. Nguyen, B. Wünsche, P. Delmas, and C. Lutteroth. 3d models from the black box: Investigating the current state of image-based modeling. 2012.

[24] J. M. Prewitt. Object enhancement and extraction. *Picture processing and Psychopictorics*, 10(1):15–19, 1970.

[25] L. A. K. Pujari and P. Reddy. Linear octrees by volume intersection. *Computer Vision, Graphics, and Image Processing*, 45(3):371–379, 1989.

[26] X. Ren. Multi-scale improves boundary detection in natural images. In *Computer Vision–ECCV 2008*, pages 533–545. Springer, 2008.

[27] L. G. Roberts. *Machine perception of three-dimensional soups*. PhD thesis, Massachusetts Institute of Technology, 1963.

[28] H. Rüther, J. Smit, and D. Kamamba. A comparison of close-range photogrammetry to terrestrial laser scanning for heritage documentation. *South African Journal of Geomatics*, 1(2):149–162, 2014.

[29] S. Singh. *PERD Iceberg Database for the Grand Banks Region*. National Research Council Canada, 1998.

[30] G. Timco. Compilation of iceberg shape and geometry data for the grand banks region. *PERD/CHC Report*, pages 20–43, 1999.

[31] G. Timco. Grand banks iceberg management. *PERD/CHC Report*, pages 20–84, 2007.

[32] G. W. Timco and N. Porters Lake. Techniques for determining the maximum draft of an iceberg. *National Research Council of Canada, Report*, pages 20–40, 2000.

[33] M. Zhou, R. Bachmayer, and B. deYoung. Working towards seafloor and iceberg mapping with slocum underwater glider. *Oceanic Engineering Society- IEEE AUV, Mississippi*, pages 1–5, 2014.

Appendix A

Surface Imaging Code

A.1 Matlab Code

A.1.1 roi.m

```
\singlespace
%%%Region of interest (manual contour finding)%%%
clear all;clc;
pitch=6;
file=dir('.../*.JPG');
for n=1:length(file)
    I=imread(['.../',file(n).name]);
    [pathstr, name, ext] = fileparts(file(n).name);
    I=I(:,:,1);
    %eval([file(n).name(1:end-4),'=temp;'])
    size=size(I);
    if size(1)>500;
    r=double(size(1)/500);
    imwrite(I(1:r:end,1:r:end),['.../',file(n).name])
    I=imread(['.../',file(n).name]);
    end
    clear size;
    I=imrotate(I,pitch);
    BW=roipoly(I);
    boundary=bwboundaries(BW);
    con=boundary{1};
    filenm=[name '_contour' '.mat'];
    save(filenm, 'con')
    plot(con(:,2),con(:,1),'r','linewidth',2);
    axis equal;
    axis off;
    saveas(gcf,[name '_contour'],'jpg');
    close(gcf);
```

```
end;exit;
```

A.1.2 gpb.m

```
%%%Hierarchical image segmentation and automatic contour finding%%%
%% 1. compute globalPb on a small image to test mex files
clear all; close all; clc;
OriFile= '.../filename.png';I=imread(OriFile);
imwrite(I(:,:,1),'.../tmp.jpg');
%% 2. change the original figure to a smaller one
I=imread('.../tmp.jpg');size=size(I);
if size(1)>250;
    r=double(size(1)/250);
    imwrite(I(1:r:end,1:r:end),'.../tmp.jpg')
    imgFile = '.../tmp.jpg';
else
    imgFile=OriFile;
end
outFile = '.../filename.mat';
gPb_orient = globalPb(imgFile, outFile);
delete(outFile);
ucm = contours2ucm(gPb_orient, 'imageSize');
figure;imshow(ucm);
ucb=im2bw(ucm,0.2);
figure;imshow(ucb);
ucb_filled=imfill(ucb,'holes');
J=ucb_filled-ucb;
boundaries = bwboundaries(J);
contour=[];
 for k=1:length(boundaries)
       b = boundaries{k};
       contour=[contour;b];
 end
figure; plot(contour(:,2),contour(:,1),'r');
axis equal;
```

The functions "globalPb" and "contour2ucm" is from the hierarchical image segmentation. [4]

A.1.3 integration.m

```
%%%Volume estimation using layer integration%%%
clear all;clc;
height=30; %m
[vertex,face]=read_off('filename.off');
%title('3D Model of Iceberg (Units in meters)');
points=vertex';x=points(:,1);y=points(:,2);points(:,3)=points(:,3)-min(points(:,3))
z=points(:,3);a=0;ratio=height/max(points(:,3));points=points.*ratio;hh=points(:,3)
for i=0:1:height-1; %round(height)
    index=find(i+1>hh& hh>i | hh==i);
    poly1=[points(index,1),points(index,2),points(index,3)];
     [X,Y]=points2contour(poly1(:,1),poly1(:,2),1,'cw');
     a=polyarea(X,Y)*2.6+a;
end
```

```
figure()
str = sprintf('Height = %f m, and volume = %f m^3',height,a);
title(str);
patch('Faces', face', 'Vertices', points,'FaceColor','w','EdgeColor','b');
axis equal;
```

A.1.4 icebergggps.m

```
%%%using GPS and distance to calculate iceberg position%%%
R=6367000;
for n=1:length(Data);
if B(n)<=0;
    micron.lat(n)=NaN;
    micron.lon(n)=NaN;
else
    micron.lat(n) = Airmarlati(n) +
    B(n)*cosd(Heading(n)+19-90)*180/(pi*R);
    micron.lon(n)= Airmarlongi(n) +
    B(n)*sind(Heading(n)+19-90)*180/pi/R./cosd(Airmarlati(n));
end
end
hold on;
plot(micron.lat,micron.lon,'*');
axis equal;
xlabel('Latitude')
ylabel('Longitude')
```

A.1.5 gps2meters.m

```
%%%change from GPS to UTM (meters)%%%
R=6367000;
for n=1:length(Data);
if B(n)<0;
    micron.lat(n)=NaN;
    micron.lon(n)=NaN;
else
    micron.lat(n) = Airmarlati(n) +
    B(n)*cosd(Heading(n)+19-90)*180/(pi*R);
    micron.lon(n)= Airmarlongi(n) +
     B(n)*sind(Heading(n)+19-90)*180/pi/R./cosd(Airmarlati(n));
    latmeter(n)=(micron.lat(n)-min(micron.lat))*111219/360*180;
    lonmeter(n)=(micron.lon(n)-min(micron.lon))*72491/360*180;
end
end
```

A.2 Octave Code

A.2.1 resize.m

```
%%%save contours to dxf for future use%%%
clear all;clc;folder= '.../';filepattern = fullfile(folder,'filename.mat');
file=dir(filepattern);
for n=1:length(file)
   [pathstr, name, ext] = fileparts(file(n).name);
   load(['.../',file(n).name]);
   theta=0; %pitch
   %height=40.51; %meter
```

```
   f_pixel=300;
   x1=con(:,2)*cosd(theta)-con(:,1)*sind(theta);
   y1=con(:,2)*sind(theta)+con(:,1)*cosd(theta);
   h_pixel=max(y1)-min(y1);
   yf=y1*f_pixel/h_pixel;
   xf=x1*f_pixel/h_pixel;
   yf=yf-min(yf);
   figure();
   plot(xf,yf);
   axis equal;
   ylim([-400,1000]);
   axis off;
   saveas(gcf,['dxf_' name],'dxf');
   close(gcf);
close;
end;exit
```

A.3 Python Code for FreeCAD

A.3.1 fc.py

```
%%%open and save to a different format%%%
import os, subprocess
from os.path import basename,splitext
root_dir = '.../'
for directory, subdirectories, files in os.walk(root_dir):
  for file in files:
    if file.endswith('.dxf'):
        base=os.path.basename(file)
        basename=os.path.splitext(base)[0];
        import importDXF
        importDXF.open(os.path.join(directory, file))
        App.setActiveDocument(basename)
        App.ActiveDocument=App.getDocument(basename)
        Gui.ActiveDocument=Gui.getDocument(basename)
        Gui.SendMsgToActiveView("ViewFit")
        __objs__=[]
        __objs__.append(FreeCAD.getDocument(basename).getObject("Polyline"))
        __objs__.append(FreeCAD.getDocument(basename).getObject("_"))
        import importDXF
        importDXF.export(__objs__,os.path.join(directory,file))
        del __objs__
     else:
        print 'helloworld'
        sys.quit()
```

A.3.2 move.py

```
%%%move all contours to the base point%%%
import FreeCAD
import importDXF
import Draft
import os, subprocess
from os.path import basename,splitext
root_dir = '.../'
for directory, subdirectories, files in os.walk(root_dir):
```

```
  for file in files:
    if file.endswith('.dxf'):
        base=os.path.basename(file)
        basename=os.path.splitext(base)[0];
        importDXF.open(os.path.join(directory, file))
        App.setActiveDocument(basename)
        App.ActiveDocument=App.getDocument(basename)
        Gui.ActiveDocument=Gui.getDocument(basename)
        Gui.SendMsgToActiveView("ViewFit")
        Gui.activateWorkbench("DraftWorkbench")
        Draft.move([FreeCAD.ActiveDocument.Polyline],FreeCAD.Vector
        (-71.2339320183,-21.6708745956,-3.5527136788e-15),copy=False)
        __objs__=[]
        __objs__.append(FreeCAD.getDocument(basename).getObject("Polyline"))
        importDXF.export(__objs__,os.path.join(directory,file))
        del __objs__
    else:
        print 'helloworld'
        sys.quit()
```

A.3.3 fcloop.py

```
%%%contour extrusion in FreeCAD%%%
import FreeCAD
import importDXF
import Part
import Mesh
import sys
import FreeCADGui
from FreeCAD import Base
import os, subprocess
from os.path import basename,splitext
root_dir = ".../"
for directory, subdirectories, files in os.walk(root_dir):
  for file in files:
    if file.endswith(".dxf"):
        base=os.path.basename(file)
        basename=os.path.splitext(base)[0];
        my_tmp_doc=FreeCAD.newDocument(basename)
        wholename=directory + file
print "directory = " + directory
print "wholename = " +  wholename
        importDXF.insert(wholename,basename)
        print os.path.splitext(os.path.basename(wholename))[0]
        list_of_segments=[]
        for obj in my_tmp_doc.findObjects("Part::Feature"):
             list_of_segments.append(obj.Shape)
        print("dbg402: list_of_segments:", list_of_segments)
print "1"
        my_dxf_face = list_of_segments[0]
print "2"
        # extrusion
        my_solid = my_dxf_face.extrude(Base.Vector(0,0,200))
        # straight linear extrusion
print "3"
        ## view and export your 3D part
        # output_stl_file=os.path.join([directory,basename],"stl")
output_stl_file = directory+basename+".stl"
```

```
print "4"
        Part.show(my_solid)
        # works only with FreeCAD GUI, ignore otherwise
        my_solid.exportStl(output_stl_file)
        print("output stl file: %s"%(output_stl_file))
        print("dbg999: end of script")
        sys.quit()
```

A.4 OpenSCAD Code

```
%%%intersection in Openscad%%%
intersection()
{
rotate([90,180,0])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_0_contour.stl");

rotate([90,180,16])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_16_contour.stl");

rotate([90,180,40])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_40_contour.stl");

rotate([90,180,56])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_56_contour.stl");

rotate([90,180,88])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_88_contour.stl");

rotate([90,180,106])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_106_contour.stl");

rotate([90,180,124])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_124_contour.stl");

rotate([90,180,140])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_140_contour.stl");

rotate([90,180,154])
translate([0,0,-100])
rotate([0,0,0])
#import(file = "dxf_154_contour.stl");

rotate([90,180,170])
translate([0,0,-100])
rotate([0,0,0])
```

```
#import(file = "dxf_170_contour.stl");
}
```

A.5 Shell Script

```
%%%link all codes above%%%
matlab -nodesktop -r "roi"
##matlab -nodesktop -r "gPb"
octave resize.m
FreeCAD fc.py
FreeCAD move.py
FreeCAD fc_loop.py
openscad build_3d.scad
matlab -nodesktop -r "integration"
```

Appendix B

LIDAR Code

B.1 vcanprocess.m

```
%%%interpret CANbus Message%%%
clear all;clc;
h = fopen('C:\Users\ywang\Downloads\lidar\18468_2.csv','r');
nextline = '';str='';nextline = fgetl(h);
[a b c d e f g i j] = strread
(nextline, '%s %s %s %s %s %s %s %s %s','delimiter', ',');
dnext=[a b c d e f g i j];
while ischar(nextline)
     if ischar(nextline)
        str = [str;dnext];
    end
    nextline = fgetl(h);
    [a b c d e f g i j] = strread
(nextline, '%s %s %s %s %s %s %s %s %s','delimiter', ',');
    dnext=[a b c d e f g i j];end
[a b c d e f g i j] = strread;
(nextline, '%s %s %s %s %s %s %s %s %s','delimiter', ',');
%hex2dec
data=zeros(size(str));
for i=1:9;data(:,i)=hex2dec(str(:,i));end
% transform
m=zeros(size(data,1),4);
for j=1:size(data,1)
    if data(j,1)==82;
        yaw=(data(j,4)*16*16+data(j,5))*0.0001*180/3.1415926;
        pitch=(data(j,6)*16*16+data(j,7))*0.0001*180/3.1415926;
        roll=(data(j,8)*16*16+data(j,9))*0.0001*180/3.1415926;
        m(j,1)=52;
        m(j,2)=yaw;
        m(j,3)=pitch;
        m(j,4)=roll;
    else if data(j,1)==48;
            lon_dec= data(j,8)*16*16*16*16*16*16+
data(j,9)*16*16*16*16+data(j,4)*16*16+data(j,5);
            lon= typecast(uint32(lon_dec),'single');
            lat_dec= data(j,6)*16*16*16*16*16*16+
data(j,7)*16*16*16*16+data(j,8)*16*16+data(j,9);
            lat= typecast(uint32(lat_dec),'single');
            m(j,1)=30;
            m(j,2)=lon;
            m(j,3)=lat;
```

```
        else data(j,1)==1;
            hour=data(j,6);
            minute=data(j,7);
            second=(data(j,8)*16*16+data(j,9))/1000;
            m(j,1)=1;
            m(j,2)=hour;
            m(j,3)=minute;
            m(j,4)=second;
        end
    end
end
save str.mat str
```

B.2 frames.m

```
%%%add up all frames to build point cloud%%%
clear all;clc;load('C:\Users\ywang\Downloads\lidar\str.mat');
%hex2dec
data=zeros(size(str));
for i=1:9;
data(:,i)=hex2dec(str(:,i));
end
% transform
m=zeros(size(data,1),4);test=zeros(size(data,1),4);i=0;
for j=1:size(data,1)
    if data(j,1)==82;
        yaw=(data(j,4)*16*16+data(j,5))*0.0001*180/3.1415926;
        pitch=(data(j,6)*16*16+data(j,7))*0.0001*180/3.1415926;
        roll=(data(j,8)*16*16+data(j,9))*0.0001*180/3.1415926;
        m(j,1)=52;
        m(j,2)=yaw;
        m(j,3)=pitch;
        m(j,4)=-roll;
    else if data(j,1)==48;
            lon_dec= data(j,2)*16*16*16*16*16*16+
data(j,3)*16*16*16*16+data(j,4)*16*16+data(j,5);
            lon= typecast(uint32(lon_dec),'single');
            lat_dec= data(j,6)*16*16*16*16*16*16+
data(j,7)*16*16*16*16+data(j,8)*16*16+data(j,9);
            lat= typecast(uint32(lat_dec),'single');
            m(j,1)=30;
            m(j,2)=lon;
            m(j,3)=lat;
            i=i+1;
            test(i,1)=30;
            test(i,2)=lon;
            test(i,3)=lat;
            test(i,4)=j;
        else data(j,1)==1;
            hour=data(j,6);
            minute=data(j,7);
            second=(data(j,8)*16*16+data(j,9))/1000;
            m(j,1)=1;
            m(j,2)=hour;
            m(j,3)=minute;
            m(j,4)=second;
        end
    end
```

```
end
file=dir('C:\Users\ywang\Downloads\lidar\test_1438003151_18468\*.csv');
data=[];
y=[];
%manually find a start point of GPS for LIDAR006 Part
%start at test L3250 m L10400 time 164200
index=zeros(length(file),8);
index(1,1)=1;
index(1,2)=1;
index(1,3)=1152489560;
index(1,4)=4630;
for i=1:1:4000;  %choose suitable frames here
    x1=importdata(['C:\Users\ywang\Downloads\lidar\test_1438003151_18468\',
file(i).name]);
    x=x1.data;
    index(i,3)=x(1,9);
    index(i,5)=round((index(i,3)-index(1,3))/(10^6));

    if index(i,5)>=60-rem(index(1,4),100);
        index(i,4)=index(1,4)+fix((index(i,5)-rem(index(1,4),100))/60)*100
+(100-rem(index(1,4),100))
        +rem((index(i,5)-rem(index(1,4),100)),60);
    else
        index(i,4)=index(1,4)+index(i,5);
    end
 list=[]; list=find(abs(fix(index(i,4)/100)-m(:,3))<0.01 & abs
(rem(index(i,4),100)-m(:,4))<0.01 & abs(m(:,1)-30<0.01));
    if isempty(list)
        index(i,1)=0;
    else
        index(i,1)=list(1);
    end
     if index(i,1)>0;
        if m(index(i,1),1)==30;
            index(i,2)=index(i,1);
        elseif m(index(i,1)-1,1)==30;
            index(i,2)=index(i,1)-1;
        elseif m(index(i,1)+1,1)==30;
            index(i,2)=index(i,1)+1;
        elseif m(index(i,1)+2,1)==30;
            index(i,2)=index(i,1)+2;
        elseif m(index(i,1)-2,1)==30;
            index(i,2)=index(i,1)-2;
        end
    else
        index(i,2)=0;
    end
    if index(i,2)>0;
        if m(index(i,2)-1,1)==52;
            yaw1=m(index(i,2)-1,2);
            pitch1=m(index(i,2)-1,3);
            roll1=m(index(i,2)-1,4);
        else
            yaw1=m(index(i,2)-2,2);
            pitch1=m(index(i,2)-2,3);
            roll1=m(index(i,2)-2,4);
        end

        if m(index(i,2)+1,1)==52;
            yaw2=m(index(i,2)+1,2);
            pitch2=m(index(i,2)+1,3);
            roll2=m(index(i,2)+1,4);
```

```
        else
            yaw2=m(index(i,2)+2,2);
            pitch2=m(index(i,2)+2,3);
            roll2=m(index(i,2)+2,4);
        end
     index(i,6)=(yaw1+yaw2)/2;
        index(i,7)=(pitch1+pitch2)/2;
        index(i,8)=(roll1+roll2)/2;
        yaw=-index(i,6)-90;
        pitch=index(i,7);
        roll=-index(i,8);
    else
        yaw=0;
        pitch=0;
        roll=0;
    end
     if index(i,1)>0;
        y=[x(1:size(x,1),1),x(1:size(x,1),2),x(1:size(x,1),3)];
        [easting, northing]=wgs2utm(m(index(i,2),3)/100,m(index(i,2),2)/100);
        REB=[cosd(yaw)*cosd(pitch)   sind(yaw)*cosd(pitch)    -sind(pitch);...
                    -sind(yaw)*cosd(roll)+cosd(yaw)*sind(pitch)*sind(roll)
cosd(yaw)*cosd(roll)+sind(roll)*sind(pitch)*sind(yaw)
                    cosd(pitch)*sind(roll);...
                    sind(yaw)*sind(roll)+cosd(yaw)*cosd(roll)*sind(pitch)
          -cosd(yaw)*sind(roll)+sind(pitch)*sind(yaw)*cosd(roll)
                    cosd(pitch)*cosd(roll)];
        y=y*REB/2;
        y(:,2)=y(:,2)+easting;
        y(:,1)=y(:,1)+northing;
        data=[data;y];
    end
end
easting1=[];northing1=[];
[easting1, northing1]=wgs2utm(test(:,3)/100,test(:,2)/100);
y=zeros(5593,3);
y(:,1)=northing1(1:5593);
y(:,2)=easting1(1:5593);
data=[data;y];
save data.txt data -ascii
```

Appendix C

C++ Code for Data Collection on BBB

C.1 lidar.cpp

```
using namespace std;
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
\section{can.cpp}
\begin{verbatim}
%%%use BBB to collect CANbus Messages%%%
using namespace std;
#include<fstream>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
FILE* CANbusStream = NULL;
char CANbusMessage[2000]="";
char ImageName[50]="";
char year[100];
char monthday[100];
char hourminute[100];
char second[100];
// Transform the information from byte to float
void IEEE754_htof(unsigned char a,unsigned char b,unsigned char c,
unsigned char d, float& val)
{ long temp=0;temp|=a;temp<<=8;temp|=b;temp<<=8;
    temp|=c;temp<<=8;temp|=d;float *p=(float *)&temp;val=*p;}

//This function is used to transform the float data to byte data for
//transmission on the CAN bus
void IEEE754_ftoh(float val,unsigned char& t1,unsigned char& t2,
unsigned char&t3,unsigned char& t4)
{ long *p=(long *)&val;
    long temp=*p; t4=temp&0xff;
    temp>>=8; t3=temp&0xff; temp>>=8; t2=temp&0xff;
    temp>>=8; t1=temp&0xff;}

void CANbus_MessageExtract(char* CANbusData)
{char id[3] = "";char dlc[1] = "";
char d1[2] = "";char d2[2] = "";char d3[2] = "";
char d4[2] = "";char d5[2] = "";char d6[2] = "";
char d7[2] = "";char d8[2] = "";
id[0] = CANbusData[8];
```

```
id[1] = CANbusData[9];
id[2] = CANbusData[10];
dlc[0] = CANbusData[15];
if (dlc[0]>=1)
{d1[0]=CANbusData[19];
d1[1]=CANbusData[20];}
if (dlc[0]>=2)
{d2[0]=CANbusData[22];
d2[1]=CANbusData[23];}
if (dlc[0]>=3)
{d3[0]=CANbusData[25];
d3[1]=CANbusData[26];}
if (dlc[0]>=4)
{d4[0]=CANbusData[28];
d4[1]=CANbusData[29];}
if (dlc[0]>=5)
{d5[0]=CANbusData[31];
d5[1]=CANbusData[32];}
if (dlc[0]>=6)
{d6[0]=CANbusData[34];
d6[1]=CANbusData[35];}
if (dlc[0]>=7)
{d7[0]=CANbusData[37];
d7[1]=CANbusData[38];}
if (dlc[0]>=8)
{
d8[0]=CANbusData[40];
d8[1]=CANbusData[41];
}
unsigned char ID = strtol(id,NULL,16);
unsigned char DLC = strtol(dlc,NULL,16);
unsigned char DATA1 = strtol ( d1, NULL, 16);
unsigned char DATA2 = strtol ( d2, NULL, 16);
unsigned char DATA3 = strtol ( d3, NULL, 16);
unsigned char DATA4 = strtol ( d4, NULL, 16);
unsigned char DATA5 = strtol ( d5, NULL, 16);
unsigned char DATA6 = strtol ( d6, NULL, 16);
unsigned char DATA7 = strtol ( d7, NULL, 16);
unsigned char DATA8 = strtol ( d8, NULL, 16);
float f;
char c[1000];
switch (ID)
{case 0x001:  // UTC
{long temp=0;
CANbusMessage[0]='\0';
temp|=DATA1;temp<<=8;temp|=DATA2;
sprintf(year, "%u/", temp);
sprintf(monthday, "%u/%u/", 2,DATA3,2,DATA4);
sprintf(hourminute, "%u/%u/", 2,DATA5,2,DATA6);
temp=0;
temp|=DATA7;temp<<=8;temp|=DATA8;
sprintf(second, "%u/%u ",2,temp/1000,3,temp-temp/1000*1000);
strcpy(c, year);
strcat(c, monthday);
strcat(c, hourminute);
strcat(c, second);
strcat(CANbusMessage,c);
strcpy (ImageName,CANbusMessage);
break;
}
```

```
case 0x030: // DGPS longitude and latitude
{
CANbusMessage[0]='\0';
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"ID %X %f ",ID,f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c,"%f ",f);
strcat(CANbusMessage,c);
break;
}
/*case 0x031: // DGPS SOG and COG
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x032: // DGPS date
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x040: // AHRS X acceleration and X angular rate
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x041: // AHRS Y acceleration and Y angular rate
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x042: // AHRS Z acceleration and Z angular rate
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
```

```
}*/
case 0x043: // AHRS X orientation and X displacement
{
CANbusMessage[0]='\0';
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"ID %X %f ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c,"%f ", f);
strcat(CANbusMessage,c);
break;
}
case 0x044: // AHRS Y orientation and Y displacement
{
CANbusMessage[0]='\0';
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"ID %X %f ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c,"%f ", f);
strcat(CANbusMessage,c);
break;
}
case 0x045: // AHRS Z orientation and Z displacement
{
CANbusMessage[0]='\0';
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"ID %X %f",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f\n", f);
strcat(CANbusMessage,c);
break;
}
/*case 0x046: // AHRS Z orientation and Z displacement
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x047: // AHRS Z orientation and Z displacement
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x048: // AHRS Z orientation and Z displacement
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
```

```
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x049: // AHRS Z orientation and Z displacement
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x04A: // AHRS Z orientation and Z displacement
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x04B: // AHRS Z orientation and Z displacement
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
case 0x04C: // AHRS Z orientation and Z displacement
{
IEEE754_htof(DATA1,DATA2,DATA3,DATA4,f);
sprintf(c,"(ID: %X) %f; ",ID, f);
strcat(CANbusMessage,c);
IEEE754_htof(DATA5,DATA6,DATA7,DATA8,f);
sprintf (c," %f; ", f);
strcat(CANbusMessage,c);
break;
}
default:
{
break;
}*/
}
}
std::string data_SaveDataToTextFile(std::string filename, std::string data)
{std::ofstream file;
file.open(("/media/660D-519D" + filename).c_str(), ios::out | ios::app);
if(!file)
{return "Could not open file";}
file << data + "";file.close();
return "saved";}
void CANbus_open()
{
std::string setupCANbus = "echo BB-DCAN0 > /sys/devices/bone_capemgr.9/slots;
```

```
sudo modprobe can;
sudo modprobe can-dev; sudo modprobe can-raw;
 sudo ip link set can0 up type can bitrate 1000000;
sudo ifconfig can0 up";
system(setupCANbus.c_str());
}
void CANbus_close()
{
pclose(CANbusStream);
}
void CANbus_Receive()
{
char CANbusData1[50]="";
char CANbusData2[50]="";
char CANbusData3[50]="";
char CANbusData4[50]="";
char CANbusData5[50]="";
// receive data
string cmd  = "candump -T 1200 -n 5 can0,001:7ff,030:7ff,043:7ff,044:7ff,045:7ff";
fgets (CANbusData1, sizeof(CANbusData1), CANbusStream);
printf("CANbus: %s\n", CANbusData1);
CANbus_MessageExtract(CANbusData1);
printf("%s\n",CANbusMessage);
data_SaveDataToTextFile("/CANdata.txt",  CANbusMessage);
fgets (CANbusData2, sizeof(CANbusData2), CANbusStream);
printf("CANbus: %s\n", CANbusData2);
CANbus_MessageExtract(CANbusData2);
printf("%s\n",CANbusMessage);
data_SaveDataToTextFile("/CANdata.txt",  CANbusMessage);
fgets (CANbusData3, sizeof(CANbusData3), CANbusStream);
printf("CANbus: %s\n", CANbusData3);
CANbus_MessageExtract(CANbusData3);
printf("%s\n",CANbusMessage);
data_SaveDataToTextFile("/CANdata.txt",  CANbusMessage);
fgets (CANbusData4, sizeof(CANbusData4), CANbusStream);
printf("CANbus: %s\n", CANbusData4);
CANbus_MessageExtract(CANbusData4);
printf("%s\n",CANbusMessage);
data_SaveDataToTextFile("/CANdata.txt",  CANbusMessage);
fgets (CANbusData5, sizeof(CANbusData5), CANbusStream);
printf("CANbus: %s\n", CANbusData5);
CANbus_MessageExtract(CANbusData5);
printf("%s\n",CANbusMessage);
data_SaveDataToTextFile("/CANdata.txt",  CANbusMessage);
}
int main()
{
char command[100]; CANbus_open();
while(1)
{
CANbus_Receive();
if (!second%10)
{
sprintf(command,"wget 'http://192.168.0.100/cgi-bin/encoder?USER=Admin
&PWD=123456&SNAPSHOT=N640x480,100&DUMMY=1' -O %s.jpg",
ImageName);
system(command);
}
}
```

```
return 0;
}
```

C.2 idar.cpp

```
using namespace std;
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>

void Capture()
{
std::string Capture_velodyne = "ifconfig eth0 up;
tshark -w /media/660D-519D/test1.pcap host 192.168.0.201";
system(Capture_velodyne.c_str());
}
int main()
{Capture();}
```

C.3 lrf.cpp

```
using namespace std;
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <iostream>
void LRF_log()
{
std::string LRF_logging = "echo BB-UART1 > /sys/devices/bone_capemgr.9/slots;
stty -F /dev/ttyO1 460800;
cat /dev/ttyO1 > /media/660D-519D/range.log";
system(LRF_logging.c_str());
}

int main()
{LRF_log();}
```